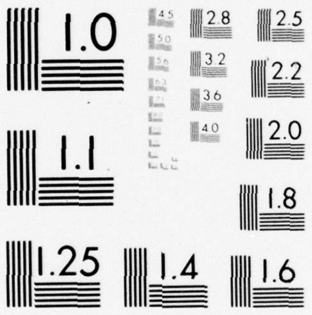
AD-A039 721 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 9/2
DESIGN OF A FLOPPY DISK DRIVE CONTROLLER INTERFACE FOR THE 6507--ETC(U)
MAR 77 M J VARNER
UNCLASSIFIED AFIT-GCS/EE/77-2 NL

1 OF 3
ADA039721


1.0
4.5
5.0
5.6
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

UNITED STATES AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY
Wright-Patterson Air Force Base, Ohio

1

GCS/EE/77-2

DESIGN OF A FLOPPY DISK DRIVE CONTROLLER INTERFACE FOR THE 65070-11 MULTIPLE TERMINAL CONTROLLER

THESIS

GCS/EE/77-2 Michael J. Varner
Capt USAF

Approved for public release; distribution unlimited

14 AFIT-GCS/EE/77-2

6

DESIGN OF A FLOPPY DISK DRIVE CONTROLLER INTERFACE FOR THE 65070-11 MULTIPLE TERMINAL CONTROLLER.

9 Master's THESIS,

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

by

10

Michael J. Varner, B.S.E.E.
Capt USAF

Graduate Computer Systems

11 18 March 1977

12 249p.

Preface

This report summarizes the progress made towards developing, implementing, and realizing an interface design which would give the MTC minicomputer secondary-storage via a magnetic disk drive and hence a stand-alone capability. Since it was not possible to test out the design in a stand-along configuration, the report discusses in depth the system development process and the testing of the hardware. It is my belief that the interface designed can be successfully applied to provide the stand-alone capability desired. In particular, the hardware developed in Chapter VI should be able to be successfully implemented into the operating system. The final task required for system integration, but which is not included in this report, would be to develop the software, I/O driver program. The report is written for a person who does not have background knowledge of either the Multiple Terminal Controller or Diskette Operating System facilities.

Throughout the course of this project, I became indepted to many without whose support the task would have not been feasible in the time allotted. Primarily, I wish to thank Captain Michael C. Wirth for his support and guidance throughout my research effort and in the development of the system design. I am also indebted to Captain Wirth for his effort and assistance in the development of the MTC PC board art work and the Avionics Laboratory, Prototyping and Standardization Group, for their support and assistance in the construction of the PC boards. Without the help of these people, the PC boards could not have been produced. I also wish to thank Dr. Gary B. Lamont for his professional assistance and guidance throughout the project and Second Lieutenant Stephen E. Smith for his professional assistance in teaching me the APPLICON Computer Aided Graphics System. I would like to thank Mr. Joseph E. Piker and the AFAL/AMFZC/Zone Shop for helping me construct the special disk controller enclosure, and for his effort in helping me obtain the necessary materials, I wish to extend my appreciation to Master Sergeant Joe L. Dyer (AFLC/DAUS). Finally, I extend special thanks to three very courageous and patient supporters to whom I am most indebted: my wife and two sons who have endured and perservered through the past several months of graduate study.

Michael J. Varner

Contents

List of Figures

List of Tables

Abstract

The 1970's was to be the decade when AFLC would completely automate its transactions on a "real time" basis through a program called the Advanced Logistics System. However, the system was never completed due to implementation difficulties. A part of that system was the Multiple Terminal Controller, which is a minicomputer, but which lacks storage capability to be used for most Air Force Applications. Based on the desire to use the Multiple Terminal Controller as a stand-alone system, this report develops an interface design for a floppy disk drive which will give the MTC secondary-storage and hence a stand-alone capability.

The design of the interface is presented which encompasses the three general functions of I/O control, direct-memory access, and program interrupt. The design is fully documented in register transfer language and other logic expressions which are finally used to realize and implement the design. While the software used to test the design is presented, the report does not discuss the software required to integrate the design into an operating system.

I. INTRODUCTION

The electronic computer of today enables a man to control his business and to assess its environment with incomparable effectiveness. By 1970, computer hardware had been developed which economically allowed minicomputers to communicate with low-speed input/output (I/O) peripheral devices. This development ushered in the beginning of putting businesses and systems "on line" (i.e., control lines are connected directly to the computer) and in "real time," (i.e., promptly and abundantly enough to control the circumstances they describe while those circumstances are developing) (Ref 1:5-7). One system which would have used this concept was the Advanced Logistics System (ALS), but the project was never completed. It was from that project that the following thesis stems.

Background

In the 1960's the Air Force Logistics Command (AFLC) conceived, approved, and finally bought a special computer system from the Control Data Corporation which would automate all the transactions of AFLC and provide the results of those transactions to users on a "real time" basis. The system was called the Advanced Logistics System. Unfortunately, due to implementation difficulties and changing system requirements, the computer system was not implemented.

Nevertheless, a subsystem of the main system called a Multiple Terminal Controller (MTC) appears to have a usefulness. The original purpose of the MTC was to function as a remote terminal controller only, but with a level of intelligence to make it adaptable to many different peripheral devices and many different configurations.

A study conducted on the MTC has revealed that it is a minicomputer. Consequently, this gives the MTC a range of additional applications. In particular, the range of applications for the MTC extend from that of an interactive remote terminal (which has already been implemented) to that of a remote batch terminal to that of a stand-alone system. Of course, a stand-alone capability would open the door to literally thousands of applications.

The major problem with the MTC is the fact that it does not have enough storage capability to be used for most Air Force applications. While the MTC memory can be extended to 64K words, it presently has only 4K words of memory. In order to be useful for the required applications, it needs a secondary-storage device such as a magnetic disk drive. Therefore, the problem is to design an interface which will allow the MTC to use the equivalent storage capability. This problem should be solved because it can save the Air Force millions of dollars in future computer requirements. There are presently many requests for stand-alone computer systems. By transforming the hundreds of available MTC systems into stand-alone systems, millions of dollars could be saved through cost avoidance.

Problem Statement

Upon evaluating the present input/output functions of the MTC system, one finds that a broader, more versatile input/output system that would take full advantage of the capability of the minicomputer system, is very much needed. Presently, the operation of the MTC is limited to that of an interactive terminal. Consequently, the full range of the capability of the MTC is not being utilized. One of the main reasons for this is that the MTC lacks secondary-storage. If secondary storage was added to this system, along with the necessary software, then the MTC could function as a stand alone system which would improve its usefulness.

This thesis investigates the feasibility of utilizing the MTC as a stand-alone system. The MTC is limited as a stand-alone system because of the lack of secondary-storage devices. Depending on the data rate and capacity desired, a number of different devices could be interfaced to the MTC. These include: a cassette tape unit, a standard magnetic tape drive, a floppy disk, a cartridge disk or a disk pack drive.

Since a floppy disk is available, economical, and will provide adequate secondary storage, it has been chosen as the secondary storage device to be interfaced. Hence, the objective of this thesis is to design and implement an interface which will provide the MTC with secondary storage. Obviously, there are many aspects involved in the development of an I/O facility that would fulfill the requirements mentioned above.

Problem Solution Approach

The following paragraphs will alert the reader to the main design considerations.

I/O Structure. Before reaching a decision as to how the communication lines of the 65070-11 Multiple Terminal Controller can be used for program control of a Floppy Disk Drive, the following must be determined:

- the function of each communication line of the MTC and the effect of the instructions set on those lines;

- the function of each communication line of the MTC and the effect of the instruction set on those lines;

- the function of each communication line of the Floppy Disk Drive (FDD);

- which lines will be used for data transfer and which will be used for control.

-the electrical parameters of each of the communication line.

<u>Direct Memory Access</u>. In order to determine whether a direct-memory access capability can be included in the I/O system, one must consider:

- whether the necessary inputs and outputs are provided by the MTC processor;

- whether the read and write circuitry can be accessed during program execution without conflicting with the operation of the processor;

- whether the speed differential between the MTC and FDD are within a compatible range.

<u>Program Interrupt</u>. Finally, investigation of the feasibility of a program interrupt capability will include the following points of consideration:

- whether the control unit can be forced to transfer out of one program into another;

- whether the existing instruction set must be modified to accomodate a program interrupt capability;

- the system impact of any required modifications.

The top-down design methodology will be used for the design. Once the three main design areas are understood, this knowledge will be used to develop the symbolic design. Next, a register transfer language description will be generated from the symbolic design. Then, from the register transfer language the hardware logic will be developed. Lastly, the hardware design will be physically realized, tested, and implemented into an operating system.

Scope of the Design Study and Realization

As mentioned earlier, there are many devices which could be interfaced to the MTC to provide the necessary secondary-storage capability. Of course, some of the devices are more difficult and expensive to interface. However, this project will interface a floppy disk drive unit due to its availability. The particular unit that will be interfaced was built by Control Data for Intel Corporation's Microcomputer Development System (MDS). The disk controller that is used was developed by Intel Corporation.

Chapter II provides some specification details and identifies the constraints which must be considered to develop the design. Chapter III provides important information about the MTC to promote continuity of the design while Chapter IV provides necessary disk information. Chapter V documents the symbolic design in register transfer language. It also explains the symbolic design using block diagrams while defining the control and data signals. Finally, Chapter VI discusses the implementation and realization of the hardware design in detail as well as covering the software developed in order to test the hardware.

Convention

The literature explaining the MTC uses a bar (¬) over a menmonic to indicate that a signal is active low while the literature explaining the DOS used a slash (/). For this report and to make typing easier, whenever a signal is active–low, its menmonic will be followed by a slash; for example, IO Reset/ means that the level on that line will be low when the I/O Reset signal is true (active). If the signal is subsequently inverted, thus making it active–high, the slash is omitted; for example, IO Reset means that the level on that line will be high when the I/O Reset signal is true.

Summary

This chapter has very briefly provided the reader with the background and objectives of this thesis. The main objective is to design a floppy disk interface for the MTC minicomputer system. In order to accomplish this objective, the three specific design areas of I/O, DMA, and interrupt must be diligently researched and applied.

II. DESIGN SPECIFICATION AND ORGANIZATION

The principle basis for the specifications outlined in this chapter is to identify the major constraints imposed upon the designer and to identify the operating characteristics desired of the interface design. The specifications are not intended to fully define the design but rather will be used to develop the design in the most expedient manner. The details will emerge as the logic structure and hardware requirements for the design are considered.

General Criteria

All additions to the MTC system should be made with the objective of at least maintaining the present processing speed since a decrease in the speed of the processor would be detrimental to the system. The design should be operationally streamlined so as to require minimal processing effort on the part of the CPU. Parallel data transfer and parallel data processing will be used whenever possible because it is faster than serial operations and faster operation generally means greater computing power. By taking this approach, the ability of the total system to perform its designed functions may be effectively increased.

In addition to the above considerations, provisions must be included to allow the future expansion of the I/O system. In particular, the interface design must support more than one peripheral device because inevitably one device is not enough. On the basis of data storage alone, the interface design must be able to support more than one peripheral device.

A primary point of emphasis will be noise reduction throughout the system in order to promote signal integrity. Namely, cable runs and discrete component leads will be kept as short as possible. This further means that discrete components should be placed as close as possible to the integrated circuit chips they support. In addition, to insure uniform transmission line signals, the transmission line signals will be conditioned utilizing an appropriate method.

The MTC and DOS systems were designed utilizing TTL logic. Hence, the logic hardware should consist of TTL-integrated-circuit logic mounted on modifiable circuit boards. Utilizing any other type of logic, for example ECL, would make the design unnecessarily more difficult and more expensive because ECL is not directly compatible to TTL. A modifiable printed circuit board which can be directly connected to the MTC bus should be developed as this feature will reduce noise and greatly simplify system interconnection. If this feature is not developed, then some type of arrangement must be developed to interconnect with the backplane of the MTC. An arrangement that will be obviously more difficult to implement. Finally, a modifiable circuit board may then be used for possible future modifications.

Discrete components will be installed on dual-in-line discrete component mounts which will further modularize the design construction and facilitate maintenance. The overall system design will be implemented using small-scale-integration (SSI) and medium-scale-integration (MSI) technology. SSI and MSI circuits will be used over large-scale-integration (LSI) circuits because they are readily available and less expensive.

Component interconnection may be made using the point-to-point wire-wrapping technique, which not only has adequate technical characteristics, but also is easily modified. All logic supply voltages should be decoupled from the logic signal inputs at each integrated circuit package to minimize the noise level on the supply voltage. In adhering to the above criteria, the overall system will be fast, efficient, and will be easily maintained.

Signal Interface Compatibility

The compatibility between signal lines going from the interface design to the MTC and DOS must be insured. Namely, standard TTL, open-collector logic, and tri-state logic will be employed consistent with hardware design requirements since these are the types of logic employed by the DOS and MTC. Utilizing any other type of logic, for example ECL, would make the design unnecessarily more difficult and more expensive. Since tri-state logic is an inherent part of the MTC and DOS systems, tri-state logic will need to be used to insure system compatibility.

Tri-state logic is rather a new concept in TTL. Hence, Appendix J provides supplementary information on tri-state logic design implementation and its advantages.

Direct Memory Access

To relieve the processor from extensive input and output operations, a direct memory access (DMA) capability will be designed into the I/O system. The DMA facility must be capable of transferring 8-bit bytes of data. The MTC and DOS systems were both designed to manipulate 8-bit data bytes. Therefore, designing an 8-bit DMA facility verses a 16bit transfer will make the design easier.

Transmission Lines

The signal paths between the MTC and DOS must be adequately designed to eliminate crosstalk and reduce transmission line reflections. Good quality signals are essential to the transmission facility.

Data Paths

Multiple use of the same data paths is encouraged whenever possible as this will reduce the overall hardware requirements. Each potential data path avoided represents a saving of switching devices. In other words, a bus structure should be utilized because it is more economical.

System Interconnection

First of all, the system interconnection is constrained by the fact that the MTC is the minicomputer system to which a peripheral device is desired to be interfaced. Secondly, the Intell DOS floppy disk drive has been selected as the peripheral device. Therefore, as illustrated in Figure 2-1, the processing system will include the MTC with its associated peripherals, the interface design which will reside near the MTC cabinet, the disk controller which includes the Interface Board and Channel Board, and the floppy disk drive unit.

System Organization

To outline the general organization of the design, the overall system will first be partitioned into functional blocks and, if possible, each block will be further subdivided according to function performed in order to simplify the construction and testing of the design. In addition, by partitioning the system, the maintenance effort will be enhanced.

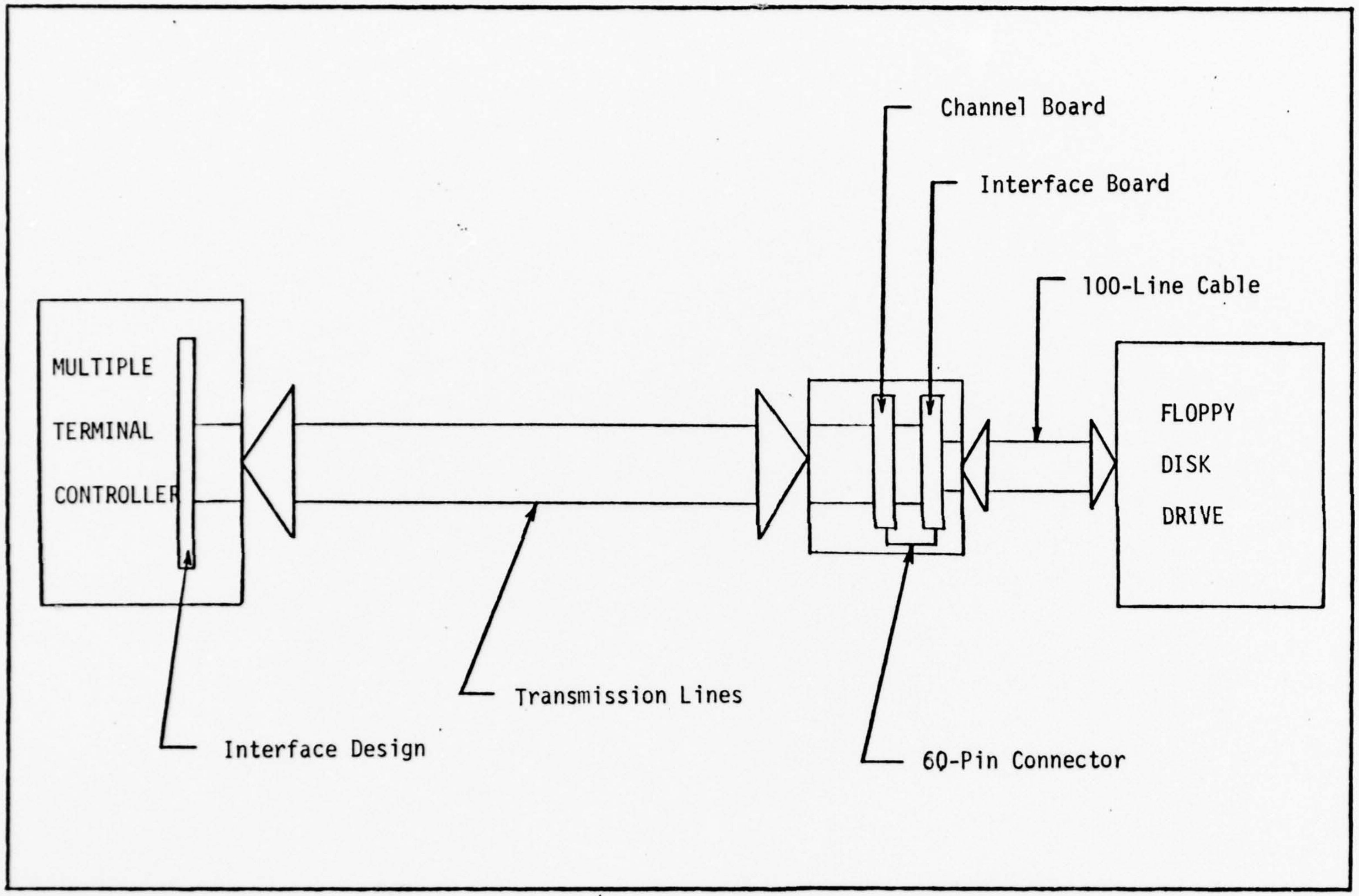


Fig. 2-1. System Interconnection Requirements

Required I/O Function

Echo Check. The interface design must have the capability of checking the integrity of the I/O system interconnection transmission line. This continuity check will include the I/O data bus.

Configuration Code. The interface design must have the capability to provide the MTC with information indicating the type of peripheral device that is communicating with the MTC. Upon request by the MTC then, the interface logic will transfer a fixed, hardwired, configuration code back to the ACU of the MTC.

Summary

In a terse manner, this chapter has presented design specifications and a method of organizing the design. The organization follows the top-down designing approach by breaking down the design into functional levels and blocks. Using parallel processing, insuring signal integrity, using modifiable circuit boards, using TTL logic, and providing direct memory access are just a few of the design requirements discussed. The plan and requirements of the interface design have been carefully laid out. Before the symbolic design is developed, however, the next two chapters will introduce the reader to the MTC and DOS systems.

III. MTC Structure

This chapter will give the reader a look at the MTC and the specific information which must be known and understood to design the controller interface. First, it should be mentioned that the minicomputer which is being interfaced is called by three different names as follows: Multiple Terminal Controller (MTC), Modular Terminal Controller (MTC), or Terminal Facility Controller (TFC). MTC for Multiple Terminal Controller will be used throughout the remainder of this report.

Overview of the MTC System

The Multiple Terminal Controller (MTC) is a modularized device which was designed as a communications coordinator for data input and output stations. The MTC is shown in Figure 3-1. Figure 3-2 and Figure 3-3 shown the subassemblies of the MTC (Ref 13:1-1, 1-5).

The MTC is composed of the following modules:

- Cabinet
- Arithmetic and Control Module (ACU)
- Input/Output Channel
- Random-Access Memory (RAM)
- Read-Only Memory (ROM)
- Operator Panel
- Programmer Panel (optional)

A block diagram of the MTC which gives the functional relationships between the modules is shown in Figure 3-4 (Ref 13:1-2). The following paragraphs describe the functions of those modules which are essential for the development of the interface design.

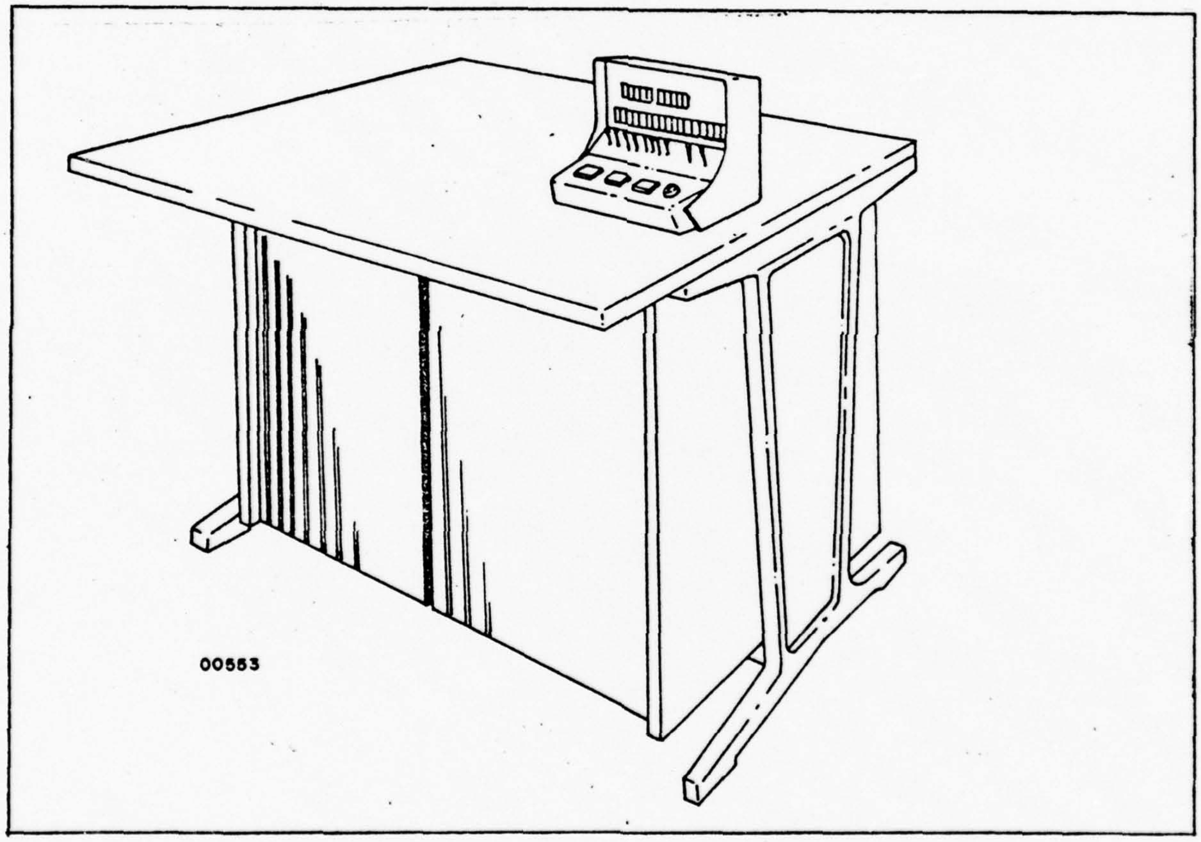


Fig. 3-1. Multiple Terminal Controller

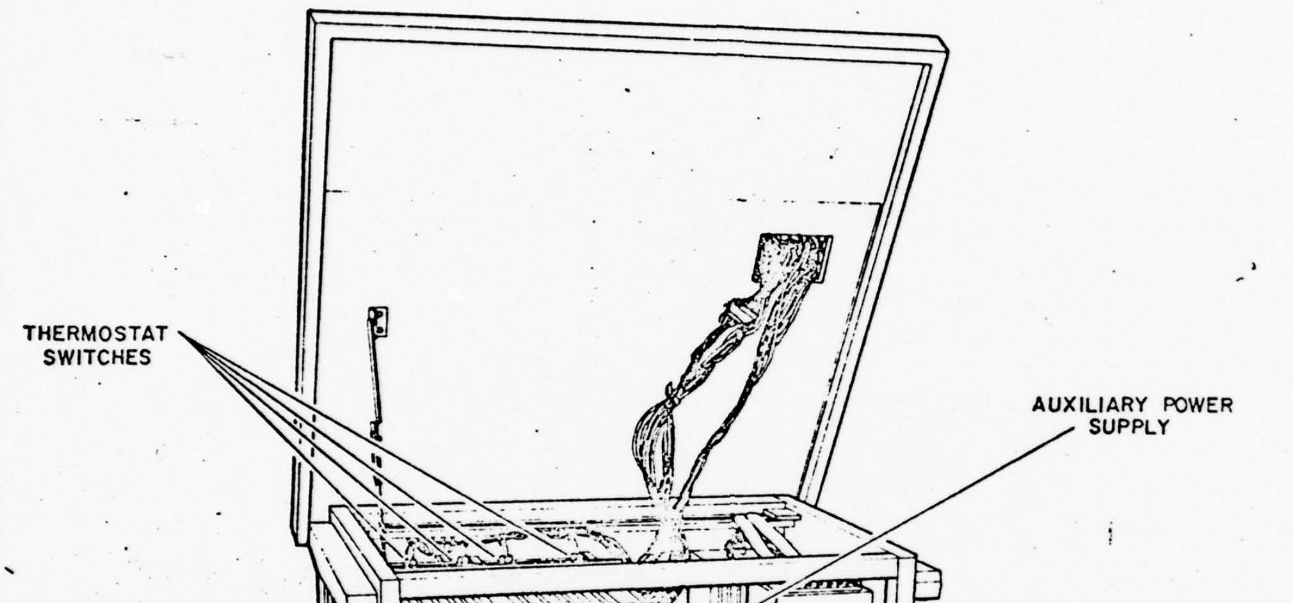


Fig. 3-2. MTC Subassemblies (Front View)

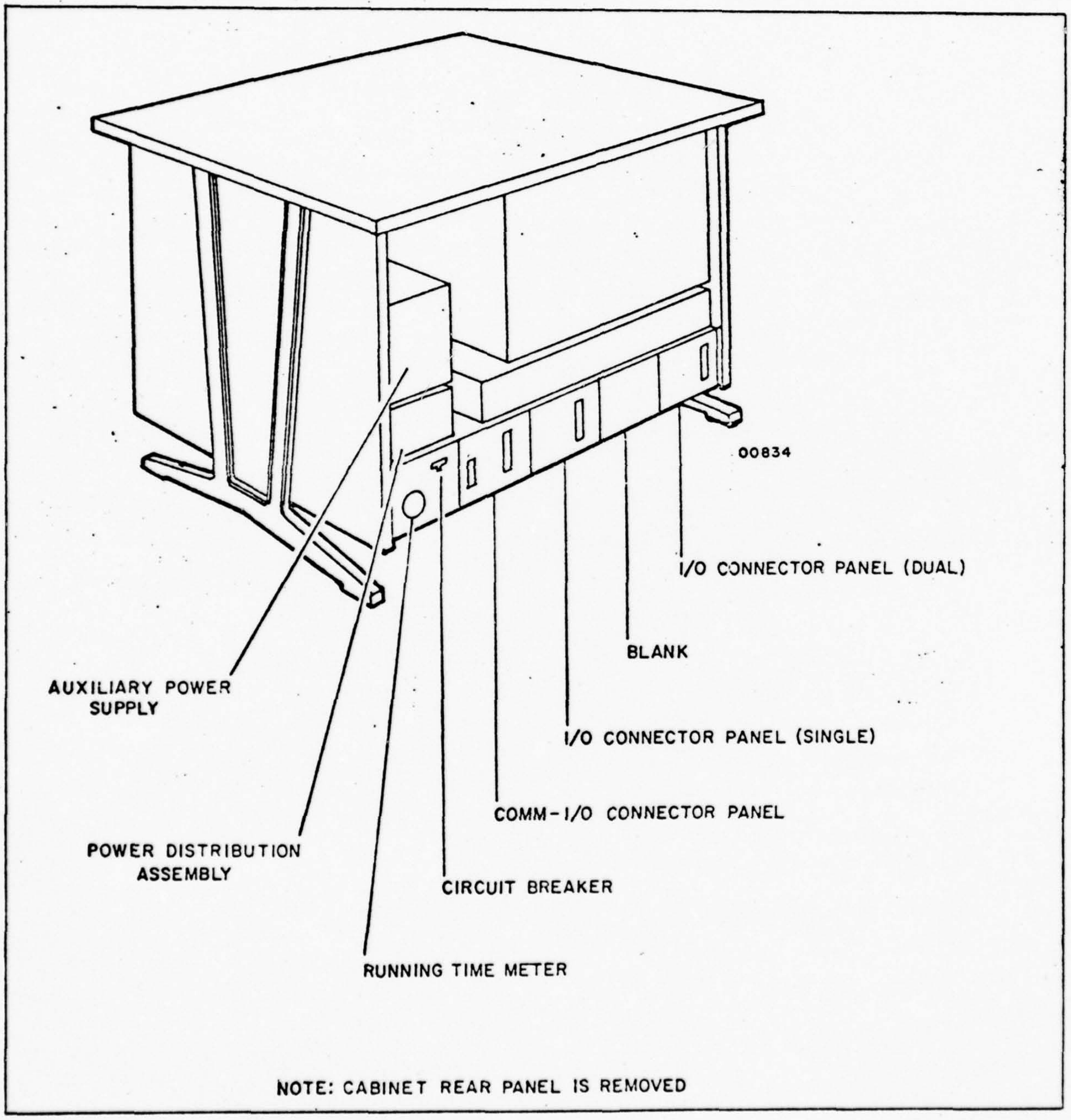

00834
I/O CONNECTOR PANEL (DUAL)
BLANK
AUXILIARY POWER SUPPLY
I/O CONNECTOR PANEL (SINGLE)
COMM-I/O CONNECTOR PANEL
POWER DISTRIBUTION ASSEMBLY
CIRCUIT BREAKER
RUNNING TIME METER
NOTE: CABINET REAR PANEL IS REMOVED

Fig. 3-3. MTC Subassemblies (Rear View)

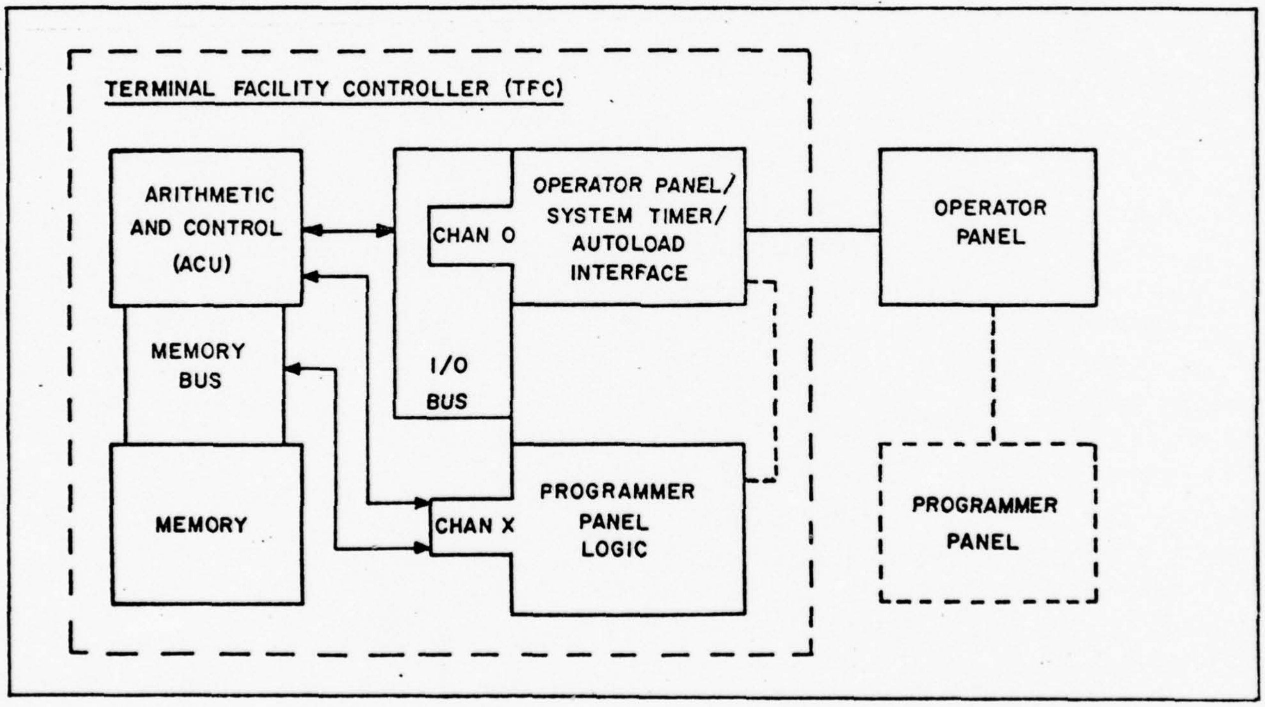

TERMINAL FACILITY CONTROLLER (TFC)
ARITHMETIC AND CONTROL (ACU)
MEMORY BUS
MEMORY
CHAN O
OPERATOR PANEL/ SYSTEM TIMER/ AUTOLOAD INTERFACE
I/O BUS
CHAN X
PROGRAMMER PANEL LOGIC
OPERATOR PANEL
PROGRAMMER PANEL

Fig. 3-4. MTC Block Diagram

Arithmetic and Control Module

The Arithmetic and Control Module (ACU) is a single-address, microprogrammed, parallel controller that uses a modifiable 32-bit instruction word and operates on an 8-bit data byte. Memory addressing can select up to 65,536 eight-bit bytes of 16,384 thirty-two-bit words of read/write or read-only memory. I/O channel addressing can select one of eight channels, and each channel can handle a maximum of 16 peripheral equipments. A channel can be an EIA RS-232-C communications interface, a direct-memory-access (DMA) peripheral interface, or a standard I/O channel peripheral interface.

The maximum channel transfer rates are as follows:

- Communication I/O channel -- 10.8K-bit transfer rate
- DMA I/O channel -- 5 MHz-byte transfer rate
- Standard I/O channel -- 900 nsec (min), 8.0 microsec (max)

Random-Access Memory (RAM)

The random/access memory (RAM) has the following important interface design characteristics: 200-nanosecond cycle time, TTL technology, 32 data lines, 14 address lines, and 6 control lines. The random-access memory modules are available as 512-RAM modules or 256-RAM modules. The 512-RAM module contains two memory banks, each consisting of 256 thirty-two-bit words. This memory is linked to the ACU through an address bus which consists of 14 address lines and a data bus which consists of 32 bidirectional lines that carry information read from memory or information to be written into memory. The six control lines include a Write Strobe signal, a Read Gate signal, and four byte-write enable lines. The 256-RAM module consists of one 256-by-32 memory bank on a single module.

RAM Memory Addressing Format

The MTC has three basic types of memory interface signals as follows: address, data, and control. Figure 3-5 illustrates these signals and the following paragraphs define their meaning.

Address: Memory Address Bits (MAB00/to MAB0D/). The 14 memory address lines select one of up to 6,144 thirty-two bit words that are available in a 12-board RAM memory. These lines carry bits that accomplish board selection, bank select, and word select functions. Figure 3-6 shows the format of the address word.

Board Select. Memory address bits 00 to 04 select the board itself, which normally consists of two RAM banks. The coding on the lines must match the settings of the address switches.

Bank Select. Memory address bit 05 selects one of the two 256-word by 32-bit RAM banks on a 512 RAM board.

Word Select. Memory address bits 06 to 0D select one of 256 storage locations in the selected RAM bank. Control signals received with the memory address will determine whether information will be read from memory or entered into memory.

Data: Memory Data (MDAT00 to MDATIF). Thirty-two memory data lines carry data to/from the RAM memory.

Read Control: Read Gate (REDGAT) Signal. The read gate (REDGAT) signal is high to gate data from the read/write memory onto the memory data lines.

Channel Interrupt Request - CIREQ/. When a device on one or more of the I/O channels requires service, the ACU receives a logical 0 on the Channel Interrupt Request/ line. The lines remains a logical 0 until the ACU (1) services the channel, (2) senses the interrupt, or (3) disables the interrupt.

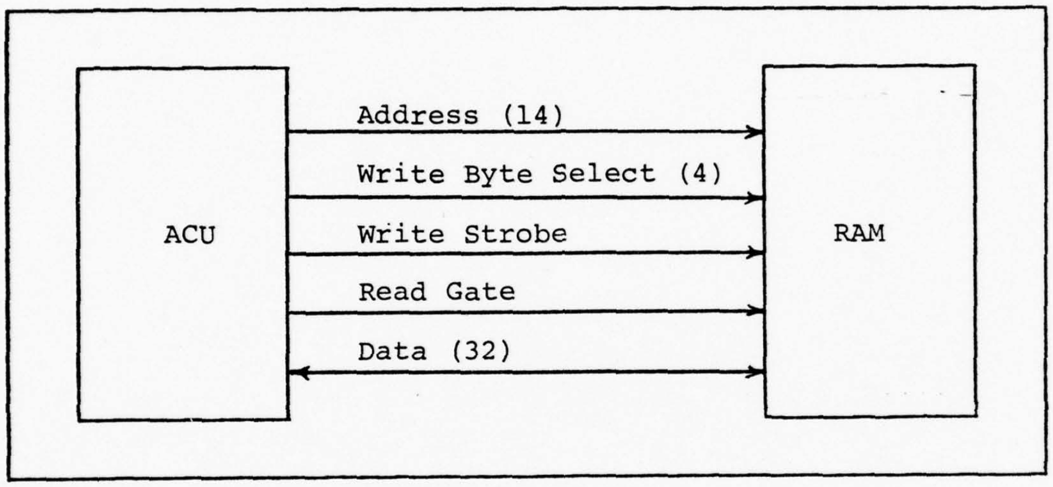


Fig. 3-5. RAM Memory Interface

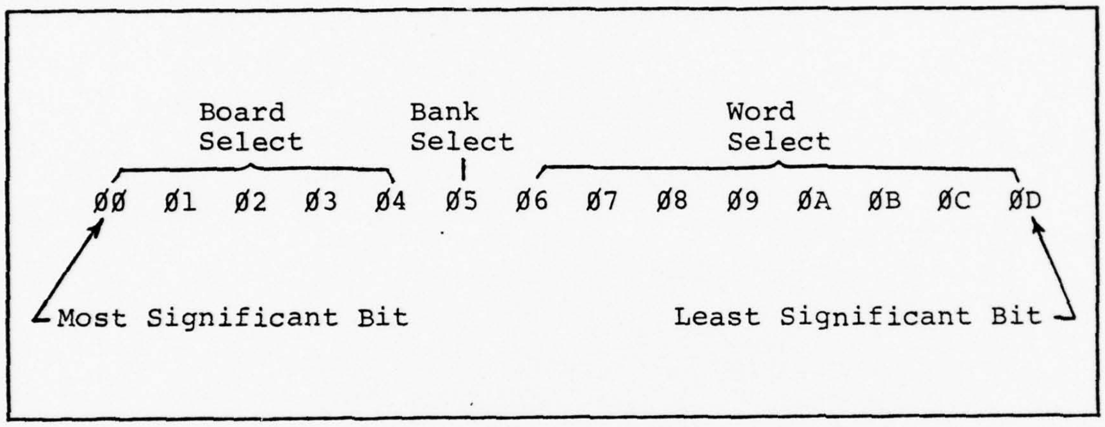


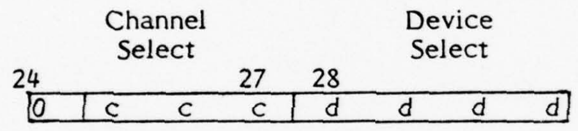
Fig. 3-6. RAM Address Word Format

I/O Control Instructions

This section will discuss the instructions which affect I/O control. (see Table I for the instruction word format.) Table II identifies each instruction by its mnemonic code, descriptive name, and hexadecimal code.

SEL. This instruction loads the contents of the Y field of the effective address, bits 24 to 31, into the I/O Select Register in the form Øcccdddd. Here, "ccc" specifies the I/O channel number (0 to 7), and "dddd" specifies the device address (0 to 16_{10}). All subsequent I/O operations occur on the selected channel and the selected device until either another SEL causes a new selection or a manual reset occurs.

Effective Address



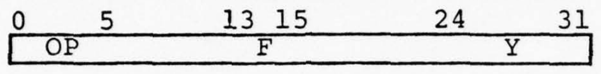
OUT. OUT transfers the contents at the effective address to the selected channel and the selected device. The function code, bits 13 to 15 of the instruction word, is applied to the peripheral device to determine the disposition of the data. (Table III lists the function codes.)

INP. This instruction transfers the byte of information that the function code, bits 13 to 15 of the instruction, specifies from the selected device on the selected channel to the effective address in memory.

TIN. TIN transfers one byte of information specified by the function code, bits 13 to 15 of the instruction, from the slected device on the channel. A logical mask operation occurs with the input byte and the effective address, and the result determines whether the next instruction executes at program address P + 1 or address P + 2.

Table I

Instruction Word Format



OP = Operation Code (Bits Ø to 5): specifies in instruction for execution (either 1B, 13, 1A, or $1C_{16}$; see Table II).

F = Function Code (Bits 13 to 15): in conjunction with the Output Line, determines the type of input or output information to be placed on the data lines (see Table III).

Y = Immediate Operand (Bits 24 to 31)

Table II

I/O Control Instructions

OP CODE (Hex)	MNEMONIC	DESCRIPTION
1B	SEL	Select Channel and Device
13	OUT	Output Transfer
1A	INP	Input Transfer
1C	TIN	Test Input

Function Codes

In conjunction with the Output Line (see Figure 3-7), the function code (bits 13 to 15 of the instruction word, Table I) determines the type of input or output information to be placed on the data lines. The function codes are listed in Table III. It is important to note that these are the functions set-up specifically for the MTC system. Since the FDD that is being interfaced is not a part of the original system design, the function codes for the FDD will differ. The exact details of the differences will be identified in a later chapter.

Interface Signals

Appendix C lists the ACU, memory, and I/O interface signals and their connector pin assignments. The following paragraphs will define and discuss each of the required memory, ACU and internal I/O interface signals essential to this design application.

Memory Interface Signals

The memory interface uses standard TTL transmission lines for its control and address lines and tri-state TTL gates to drive its two-way data lines. The following signal definitions are in three groups: address, data, and control.

Address: (Memory Address) - (MAB00 to MAB0D). The ACU uses the 14 (Memory Address) lines to select one of $16,384_{10}$ words from the RAM or ROM memory. They are active and stable 50 ns after the leading remains so for a minimum of 250 ns (see timing Figures 3-8 and 3-9).

Data: Memory Data - MDAT00 to MDATIF. Thirty-two Memory Data lines carry data to or from memory. During a memory read, these two-way lines become stable at a maximum of 150 ns after the Memory

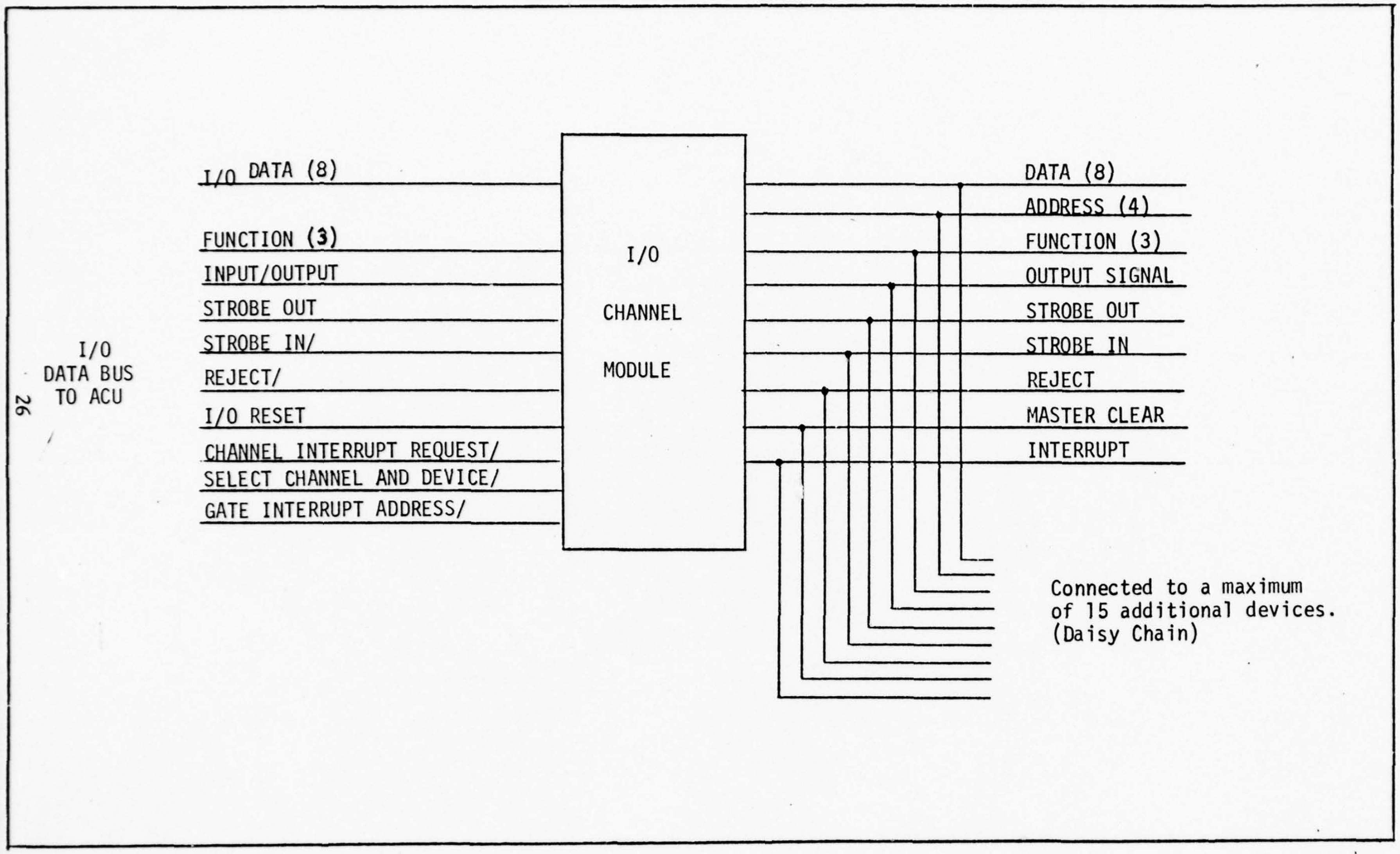


Fig. 3-7. I/O Channel Interfaces

Table III

Function Codes

OCTAL FUNCTION CODE	OUTPUT USAGE (OUTPUT LINE IS SET)	INPUT USAGE (OUTPUT LINE IS CLEAR)
0	Data	Data
1	Control 1	Status 1
2	Control 2	Status 2
3	Control 3	Status 3
4	Position 1 (X)	Position 1 (X)
5	Position 2 (Y)	Position 2 (Y)
6	Echo Check	Echo Check *
7	-	Configuration *

* Non-rejectable function; NOTE: input echo check is non-rejectable only if the previous operation was an accepted output echo check.

Address lines are stable. They remain stable until the trailing edge of the (Write Strobe) signal and remain so until 50 ns after the trailing edge of the (Write Strobe) (see Figures 3-8 and 3-9).

During a write, the 32-bit word arranges as follows:

WRTBYTØ1	WRTBYT11	WRTBYT21	WRTBYT31
ØØ ... Ø7	Ø ... ØF	1Ø ... 17	18 ... 1F

During a write, one or more of the 8-bit bytes of data enters memory for each write cycle.

Control. The following are the read/write control signals of the memory interface.

1. Write Strobe/ - WRTSTRO/. The ACU puts a logical 0 on the Write Strobe/ line to cause the memory to write new information at the addressed location. This signal gates with the Write Byte Control/ signals, is stable 150 ns after the leading edge of the Memory Port Reserved/ signal and remains so for 100 ns (see Figure 3-8).

2. Write Byte Control/ - WRTBYTØ to WRTBYT3/. The ACU puts a logical 0 on one of the four Write Byte Control lines to enable one of the four 8-bit bytes of the 32-bit word for a write. The ACU decodes the two least significant bits, $1E_{16}$ and $1F_{16}$, of the instruction word to generate one of the four Write Byte Control Signals. These lines are stable 50 ns after the beginning of an ACU write memory cycle (leading edge of the Memory Port Reserved signal) and remain so until the trailing edge of the Memory Port Reserved/ (see Figure 3-8).

3. Read Gate - REDGAT. When the ACU wants to read data from the memory, it puts a logical 1 on the Read Gate line to gate the

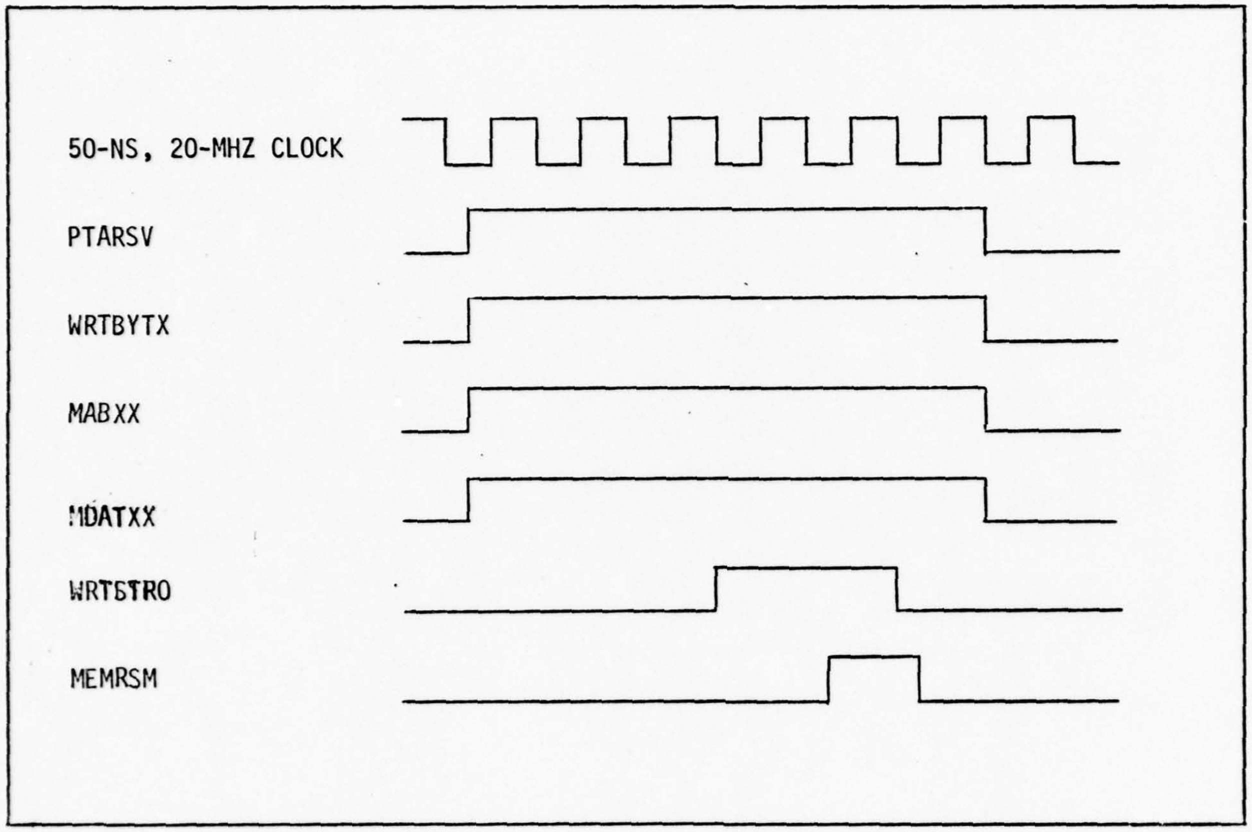


Fig. 3-8. A Write to Memory

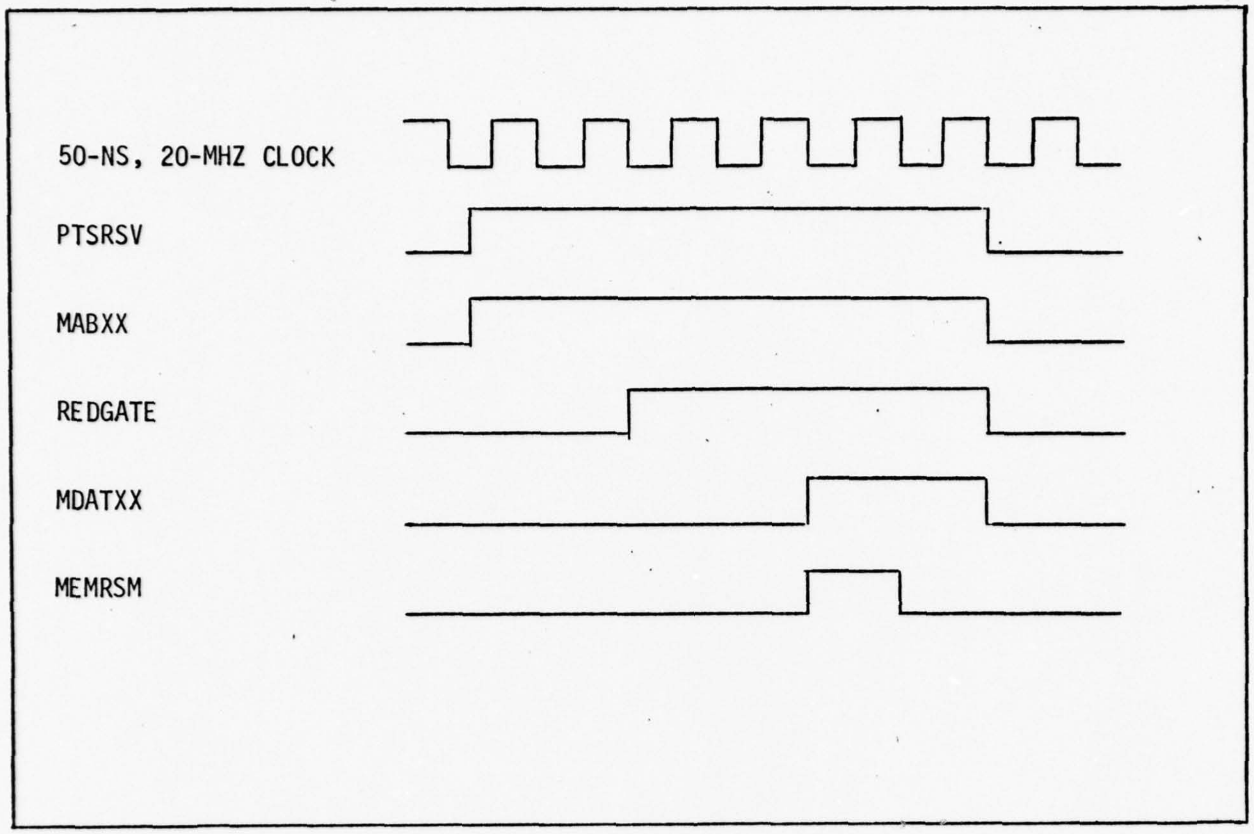


Fig. 3-9. A Read From Memory

memory data to the Memory Data lines. This signal is stable 100 ns after the beginning of an ACU memory read cycle (leading edge of the Memory Port Reserved/ signal and remains so for 200 ns (see Figure 3-9).

4. Memory Resume/ - MEMRSM. The ACU puts a logical 0 on the Memory Resume/ line to indicate at the trailing edge that either (1) requested memory data is on the Memory Data lines, or (2) data on the Memory Data Write lines for storage is now in memory. It is stable at a maximum of 200 ns after the leading edge of the Memory Port Reserved/ signal and remains so for 40 ns, minimum, to 60 ns, maximum.

5. Request Memory Part A/ - REQPTA/. Memory Port A gives an external device direct access to memory. A request for this memory has priority over an ACU or Control Panel request for Memory Port C. For subsequent requests at the maximum speed, Memory Port A and Memory Port C alternate in accessing memory. The I/O Channel puts a logical ∅ on the Request Memory Port A/ line to request access to Memory Port A. This signal remains a logical ∅ until the ACU responds with a logical ∅ on the Memory Port A Reserved/ line to signal that Memory Port A is available.

6. Memory Port A Reserved/ - PTARSV/. When an I/O channel requests access to Memory Port A, Memory Port A responds with a logical ∅ on the Memory Port A Reserved/ line to indicate that it is now available (see also "Request Memory Port A/ - REQPTA/").

Internal Input/Output ACU Interface

Control signals, data, and status are exchanged between the ACU and the I/O channels over the I/O Data Bus. Physically, this is accomplished by means of a backplane printed circuit board in the logic chassis which connects the upper board connectors (designated "P1"). TTL logic levels (+2.8 to +5.0 volts is a logical 1, and 0.0 to +0.45 volts is a logical 0) are used in this interface.

The internal I/O interface communicates separately with up to eight I/O channels. Standard TTL gates drive some of the signal lines while tri-state TTL gates drive the other (Select Channel and Device/, Fuction, I/O Data, I/O Reset/, and Gate Interrupt Address/). Signal mnemonics are as follows:

- 5 Mega Hertz Clock/ - IOCLK/
- Select Channel and Device/ - SELCH/
- Function - FUNTØ to FUNT2
- Input/Output - INOUT
- Strobe Out/ - STROBO/
- Strobe In/ - STROBI/
- Input/Output Data - IODBØØ to IODBØ7
- Reject/ - REJECT/
- I/O Reset/ - IORESET/
- Channel Interrupt Request/ - CIREQ/
- Gate Interrupt Address/ - GAITAD/

The above signals are defined in the following paragraphs.

<u>5 MegaHertz Clock/ - IOCLK/</u>. On the 5 Mega Hertz Clock/ line, the ACU suppies a 200-nsec square wave to the I/O channel.

<u>Select Channel and Device - SELCH/</u>. The ACU puts a logical 0 on the Select Channel and Device/ line to direct all channels to compare its address to the channel address on the Input/Output Data lines. Pulse width is from 40 nsec to 60 nsec. The Input/Output Data lines stabilize a minimum of 50 nsec before the leading edge of the SELCH/ signal and remain stable for 50 nsec, minimum, after the trailing edge of the signal.

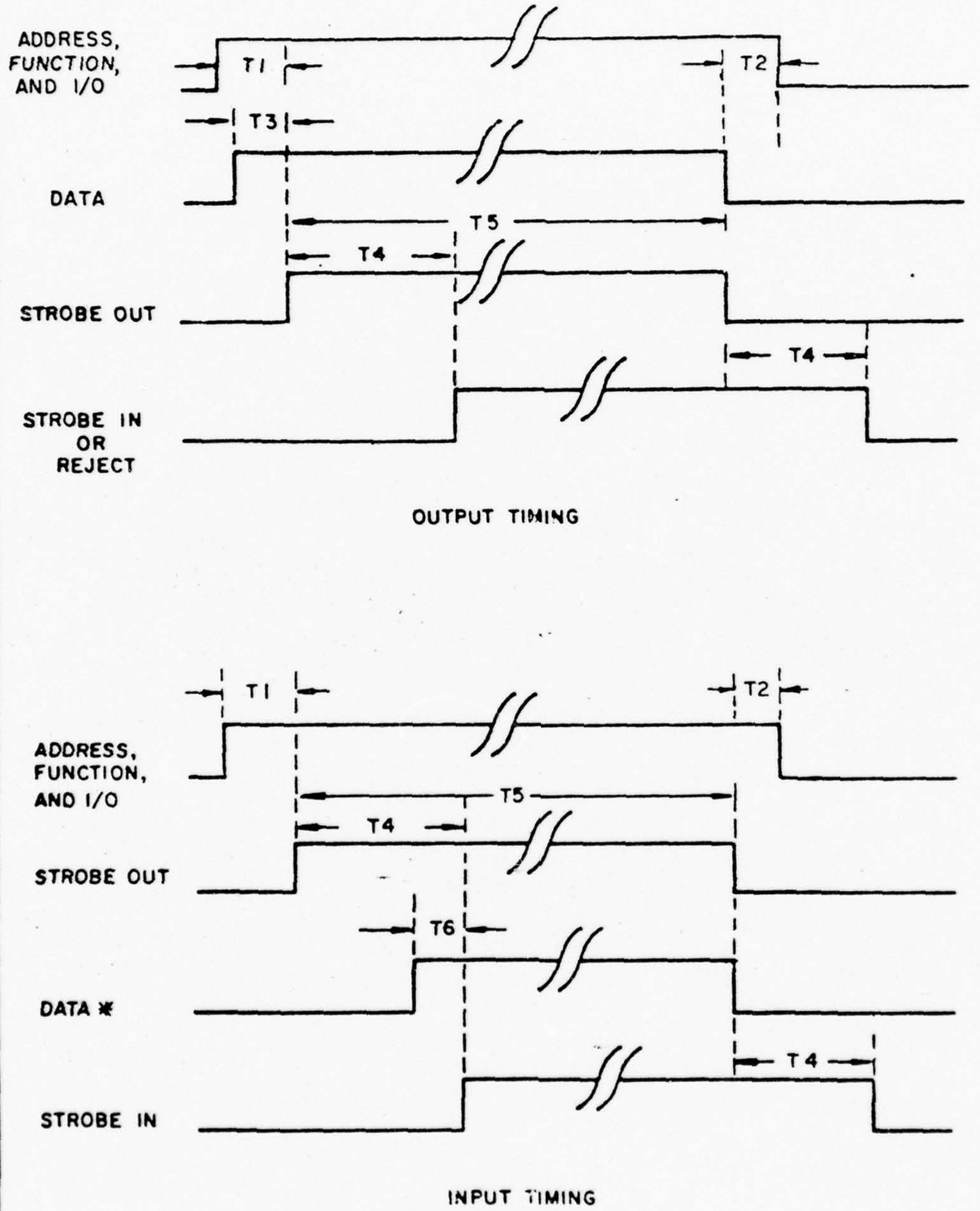
Function - FUNT0 to FUNT2. The Function lines together with the Input/Output line specify one of eight input or one of eight output functions for execution by the selected peripheral or channel, if the channel performs peripheral functions. These lines are stable 200 nsec, minimum, prior to the Strobe Out/ signal and remain stable 250 nsec, minimum, following the Strobe Out/ signal (see Figure 3-10).

For a Select instruction, the Function lines, together with the Select Channel and Device/ line, control the enabling of a channel's interrupt system, as defined for a Select instruction. In this case, Function lines one and two correspond to Select instruction bits 14 and 15, respectively. These lines are stable 50 nsec, minimum, before the leading edge of the SELCH signal and remain so for 50 nsec, minimum, after the trailing edge of this signal.

Input/Output line - INOUT. The ACU puts a logical 1 on this line to direct the selected channel to do an output function. When this line is a logical 0, the selected channel does an input function.

Strobe Out Line - STROBO/. During an output function, the ACU puts a logical 0 on this line to tell the selected channel that the Input/Output, Function, Select Channel and Device, and Input/Output Data lines are stable and that the selected channel can sample them. During an input function, the ACU puts a logical 0 on this line when it wants to sample data lines from the selected channel and device.

Strobe In Line - STROBI/. During an ACU output function, the selected channel and device puts a logical 0 on this line to tell the ACU that it accepted the information on the data lines. During an ACU input function, the selected channel and device puts a logical 0 on this line to indicate that the requested information is on the data lines and is stable.



T1 ≥ 200 ns AT THE INTERNAL I/O BUS
T2 ≥ 250 ns AT THE INTERNAL I/O BUS
T3 ≥ 150 ns AT THE INTERNAL I/O BUS
T4 ≤ 500 ns (I/O CHANNEL), ≈75 ns (COMM CHANNEL) AT THE SELECTED DEVICE
T5 ≤ 8.0 us AT THE INTERNAL I/O BUS
T6 ≥ 150 ns AT THE SELECTED DEVICE

* DATA- ANY INFORMATION PRESENT ON THE 8 DATA LINES DURING THE INPUT-FUNCTION STROBE-IN TIME.

Fig. 3-10. I/O Channel Timing

Input/Output Data Lines - IODB00 to IODB07. The eight two-way data lines (1) transmit the select channel and device code to each channel, (2) receive one 8-bit byte of information from the selected channel and device, (3) transmit one 8-bit byte of data to the selected channel and device, and (4) receive a logical 1 to identify an interrupting channel.

Reject/ - REJECT/. The selected channel and device puts a logical 0 on the Reject/ line to tell the ACU that it did not accept the function and/or the data. The Reject/ signal goes false in less that 8.0 usec after the Strobe Out/ signal goes false. It remains false until after Strobe Out/ goes true (see Figure 3-10).

Input/Output Reset - IORESET/. As a logical 0, the I/O Reset/ signal unconditionally transmits a master clear to all the I/O channels. This signal is generated during the initial power on sequence and whenever a system master clear is entered from a control panel.

Gate Interrupt Address - GAITAD/. When the ACU senses for an interrupt, it puts a logical 0 on the Gate Interrupt Address/ line to direct any interrupting channel to respond within 60 nsec with a logical 0 on the Input/Output Data line assigned to the channel (see Table IV for interrupting channel address assignments).

Table IV

Channel Address Assignments

Interrupting Channel	Assigned Input/Output Data Line
Ø (Highest Priority)	IODBØØ
1	IODBØ1
2	IODBØ2
3	IODBØ3
4	IODBØ4
5	IODBØ5
6	IODBØ6
7 (Lowest Priority)	IODBØ7

Summary

This chapter has given the reader a brief look at the MTC structure and a comprehensive explanation of the interface parameters of the MTC system. The capability of the arithmetic and control unit (ACU) and the random-access memory modules were each discussed. Each of the four I/O control instructions were explained. It has pointed out the memory and I/O control, address, and data signals which must be used to interface any peripheral device to the MTC system. It has also given a detailed description of each of the interface signals which the designer must know and understand in order to develop an interface design. With a deep understanding of the interface signals and their timing relationships as outlined in this chapter, the designer is ready to proceed to gain the same understanding of the peripheral device. The peripheral device is presented in the next chapter in order to gain the required understanding.

IV. DISKETTE OPERATING SYSTEM

Introduction

As mentioned in Chapter One, the Diskette Operating System (DOS) that is being interfaced to the Multiple Terminal Controller (MTC) System was developed by Control Data Corporation for Intel Corporation's INTELLEC Microcomputer Development System. This chapter provides a comprehensive explanation of the DOS in order for the reader to understand the interface design. The writer is indebted to Intel Corporation for much of the material in this chapter which was selected from their Diskette Operating System - Microcomputer Development System Hardware Reference Manual (Ref 4).

The INTELLEC MDS Diskette Operating System provide bulk storage capability for Intel's *INTELLEC MDS system. The Diskette Operating* System includes an intelligent controller and up to two diskette drives. Each drive provides 2,050,048 user-accessible data bits of storage with a data transfer rate of 250,000 bits/second. The controller has been implemented with Intel's powerful Series 3000 Bipolar Computing Elements. The controller provides an interface to the INTELLEC MDS bus, as well as supporting the two diskette drives. For this design, however, the controller provides an interface to the MTC bus and supports only one diskette drive. The Diskette Operating System records all data in the IBM soft-sectored format, described in Appendix I.

System Overview

In addition to one or two diskette drives, their enclosure and power supplies, the Diskette System consists of the Channel Board and the Interface Board. These two printed circuit boards normally reside in the INTELLEC MDS cabinet and constitute what is known as the diskette controller. For this design, however, the circuit boards will reside within specially designed enclosure. Each of the system components is shown in Figure 4-1, and will be described in the following paragraphs:

The Channel Board is the primary control module within the Diskette System. The Channel Board receives, decodes and responds to channel commands from a Central Processor Unit (CPU). The Channel Board can access system memory to determine the particular diskette operations to be performed and to fetch the parameters required for the successful completion of the specified operations. The Channel Board also monitors Diskette System status and error conditions, and organizes these indications into "result type" and "result byte" words that can be read by a CPU.

The control functions of the Channel and Interface Boards are provided by an 8-bit microprogrammed processor, implemented with Intel's Series 3000 Bipolar Computing Elements. The 8-bit controller includes four 3002 Central Processing Elements (2-bit slice per CPE), a 3001 Microprogram Control Unit and 512 x 32 bits of 3604 programmable-read-only-memory (PROM) which stores the microprogram. The processing and control capabilities of the Diskette system are achieved by execution of the microprogram.

The Interface Board provides the diskette controller with a means of communicating with the diskette drives, as well as with the system bus. Under control of the microprogram executed from the Channel Board, the Interface Board generates those signals which cause the read/write head on the selected drive to be loaded (i.e., to come in contact with the diskette platter), then cause the head to move to the proper track. The Interface Board accepts the data being read off the diskette, interprets certain synchronizing bit patterns, checks the validity of the data using a cyclic redundancy check (CRC) polynomial, and passes the data to the Channel Board.

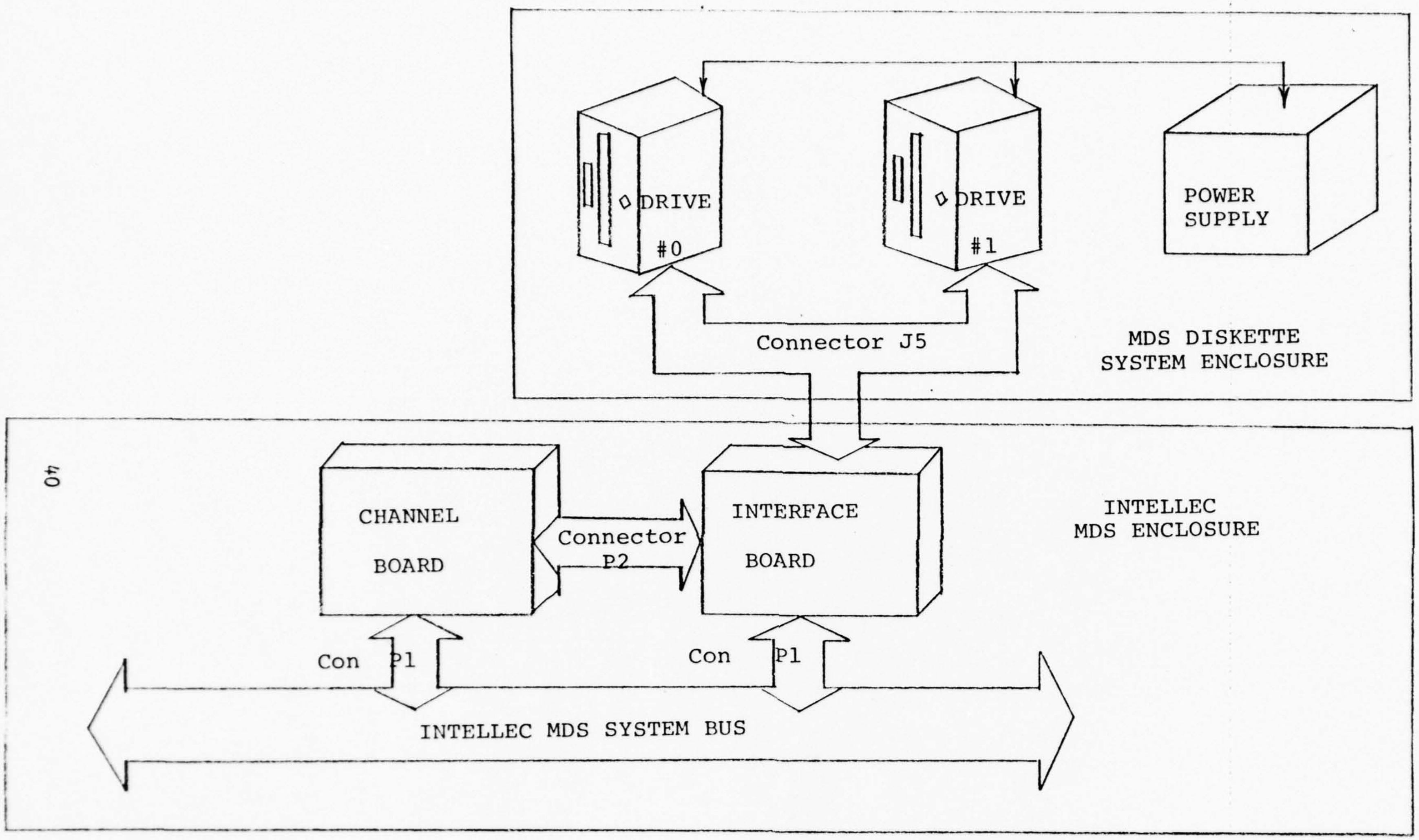


Fig. 4-1. Diskette System Block Diagram

During write operations, the Interface Board outputs the data and clock bits to the selected drive at the proper times. It also generates CRC characters which are appended to the data; this allows the data to be verified when it is subsequently read.

When the diskette controller requires access to system memory, the Interface Board requests and maintains master control of the system bus, and generates the appropriate memory command.

When a CPU issues a channel command to the Diskette System, the Interface Board acknowledges the command as required by bus protocol.

Each diskette drive consists of read/write and control electronics (on a single printed circuit board), drive mechanism, read/write head, track positioning mechanism an the removable diskette platter. These components interact to perform the following functions:

Interpret and generate control signals

Move read/write head to selected track

Read and write data

Table V lists the performance characteristics for each diskette drive.

TABLE V

Diskette Drive Performance Specifications

Capacity (formatted)	Per Disk -	256,256 bytes
	Per Track-	3,328 bytes
Data Transfer Rate:		250 kilobits/second
Access Time:	Track to Track -	10 ms
	Settling Time -	10 ms
Average Access Time:		260 ms
Rotational Speed:		360 RPM
Average Latency:		83 ms
Recording Mode:		Frequency Modulation

Operational Summary and Programming Considerations

All diskette operations are initiated by a Central Processor Unit (CPU). Once initiated, however, the Diskette Channel completes the specified operation without further intervention on the part of the CPU. From the CPU's point of view, there are only three general steps required to complete any diskette operation:

(1) The CPU must prepare and store in system memory an I/O Parameter Block (IOPB) for each operation to be performed. If multiple operations are desired, the IOPB's must be linked together in the proper order. An IOPB (ten bytes) specifies a particular diskette operation and provides all of the parameters required for execution of that operation.

(2) The CPU must then pass the memory address of the first (or only) IOPB to the Diskette Channel.

(3) The CPU must process the result information from the Diskette Channel upon completion of the operation(s).

The preparation of the IOPB(s) by the CPU, in itself, requires no interaction with the Diskette Channel. The passing of the memory address for the IOPB and the result processing, however, do require interaction. Seven channel commands have been defined to allow the CPU to perform these interactive steps. Four of the channel commands are the result of the CPU executing an output instruction to a dedicated I/O port address, while the other three commands are the result of input instructions to dedicated ports. The seven channel commands are:

(1) Write memory address lower (output)

(2) Write memory address upper and start the diskette operation (output)

(3) Stop diskette operation (output)

(4) Reset the channel (output)

(5) Read subsystem status (input)

(6) Read result type (input)

(7) Read result byte (input)

The CPU outputs the memory address of the first (or only) IOPB by executing channel commands one and two. Upon execution of channel command two, the Diskette Channel will request master control of the system bus, fetch the diskette instruction and associated parameters from the IOPB(s), and proceed to perform the specified diskette operation(s). The diskette instruction byte in the IOPB can specify any one of seven diskette operations:

(1) Recalibrate (seek track 00)

(2) Seek

(3) Format a track

(4) Write data (with data address marks)

(5) Write data (with deleted address marks)

(6) Read data

(7) Verify CRC

The Diskette Channel can interrupt the CPU when the operation (or group of linked operations) is completed or when the diskette ready status changes. The host system software can implement its CPU interrupt mechanism via this direct interrupt feature or it can "poll" the Diskette Channel by executing channel command five (read subsystem status). When the CPU determines that the operation sequence has been completed (either by receiving an interrupt request or by reading the interrupt status), the CPU should execute channel commands six and seven (read result type and read result byte) to determine whether the diskette operations were successfully completed, and if not, which type of error occurred.

Thus, in summary, certain channel commands are executed by the CPU to point the Diskette Channel to an IOPB (or group of linked IOPB's) in system memory, and initiate the operation sequence. The Diskette

Channel, then, accesses the IOPB(s) to perform the diskette operation(s) specified by the instruction byte of the IOPB(s). The Diskette Channel will, if enabled by the IOPB, generate an I/O complete interrupt request:

(1) Upon completion of each unchained diskette operation,

(2) Upon completion of the last operation in a chain of linked diskette operations, or

(3) Upon detection of an error during an intermediate operation in a chain of linked operations.

The CPU, then, executes other channel commands to determine the result(s) of the diskette operation(s).

The preceding paragraphs mentioned the channel commands, diskette operations and the IOPB without defining them explicitly. That is because up until now, the primary intention has been to identify clearly the function of each in the overall operation of the *Diskette Channel*. Subsequent sections of this chapter provide detailed information on the use and format of the channel commands, the diskette operations and the IOPB. An error section will define each of the error conditions that can be indicated when the "read result byte" channel command is executed by the CPU.

Channel Commands

There are seven channel commands to which the Diskette Channel will respond. Four of the channel commands are issued when a CPU executes output (I/O write) instructions with the appropriate eight-bit I/O addresses. The other three commands are issued when the CPU executes input (I/O read) instructions with the appropriate I/O addresses.

When the CPU executes one of the output channel commands, it activates the I/O address on address lines ADRØ/ - ADR7/ and ADR8/ - ADRF / of the system bus. Depending on the particular channel command, the CPU may also place relevant data on data lines DAT0/ - DAT7/ of the bus. The CPU maintains the data lines until the Diskette Channel returns the transfer acknowledge (XACK/) signal.

When the CPU executes one of the input channel commands, it activates the I/O read (IORC/) line and duplicates the appropriate I/O address on both halves of the bus. The CPU expects the Diskette Channel to activate the transfer acknowledge (XACK/) line when it has places the requested data on data lines DAT0/ - DAT7/.

The Diskette Channel differentiates between the different channel commands by interrogating the I/O read (IORC/) and I/O write (IOWC/) lines and the three least significant address lines (ADR0/ - ADR2/). The five most significant I/O address lines (ADR3/ - ADR7/) define the switch-selectable BASE address for the Diskette Channel.

If the Diskette Channel is not busy, it will respond to an output channel command within three microseconds. If it is busy, the "write MA lower" and "write MA upper" commands are ignored; no acknowledge is returned. (Note: Because no acknowledge is returned in this case, it could be possible to "hang up" the host system if the system does not include a Fail Safe time-out provision, as is provided on the Front Panel Control Module in the INTELLEC MDS system.) The "stop" and "reset" commands, however, are acknowledged even if the Diskette Channel is busy. "Stop" is stored and executed at the end of the current diskette operation. "Reset" is executed immediately (if issued during a data write operation, garbled data will be written).

The Diskette System responds to "read subsystem status" and "read result type" input channel commands within one microsecond. The information returned in response to a "read subsystem status" command is always valid. The eight bits of data returned in response to a "read result type" command, however, are only valid if the Diskette Channel will, if not busy, respond to a "read result byte" input command within 3 microseconds. If the Diskette Channel is busy, however, it ignores the "read result byte" command (i.e., no acknowledge is returned). The "read result type" and "read result byte" commands must be executed sequentially ("read result type" <u>first</u>), and should be executed only in response to an interrupt request from the Diskette Channel; execution at other times could produce erroneous result data.

The use and format of each of the seven channel commands is described below:

<u>Write Memory Address Lower (output)</u>. This channel command outputs the low order byte of the 16-bit memory address that points to byte one ("channel word") of the first (or only) IOPB.

System address bus:	BASE + 1
System data bus:	Eight least significant bits of the 16-bit memory address that point to the first IOPB.

<u>Write Memory Address Upper and Start the Diskette Operation (output)</u>. This channel command outputs the high order byte of the 16-bit memory address that points to byte 1 of the first (or only) IOPB. This command also causes the Diskette Channel to begin executing the diskette operation specified in byte 2 (instruction byte) of the addressed IOPB.

System address bus:	BASE + 2
System address bus:	Eight most significant bits of the 16-bit memory address that point to the first IOPB.

Stop Diskette Operation(output). This output channel command causes the Diskette Channel to cease operations after completing the current diskette operation. The Diskette Channel will not proceed to the next linked IOPB.

System address bus:	BASE + 3
System data bus:	Not used

Reset Diskette System(output). This output channel command causes all control logic in the Diskette Channel to be reset in an initialized state. If this command is issued while a "write data" diskette operation is in progress, the data in the sector currently being written will be garbled. This command is intended to clear a "hang up" in the Diskette Channel.

System address bus:	BASE + 7
System data bus:	Not used

Read Subsystem Status(input). This input channel command causes the Diskette Channel to return four status bits to the CPU. The four bits are:

(1) bit 0 - ready status of drive 0

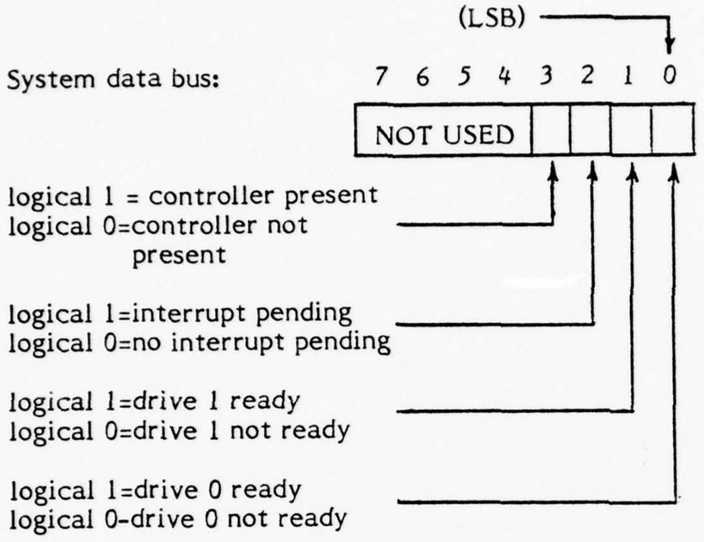
(2) bit 1 - ready status of drive 1

(3) bit 2 - state of the channel's interrupt flip-flop

(4) bit 3 - controller presence indicator

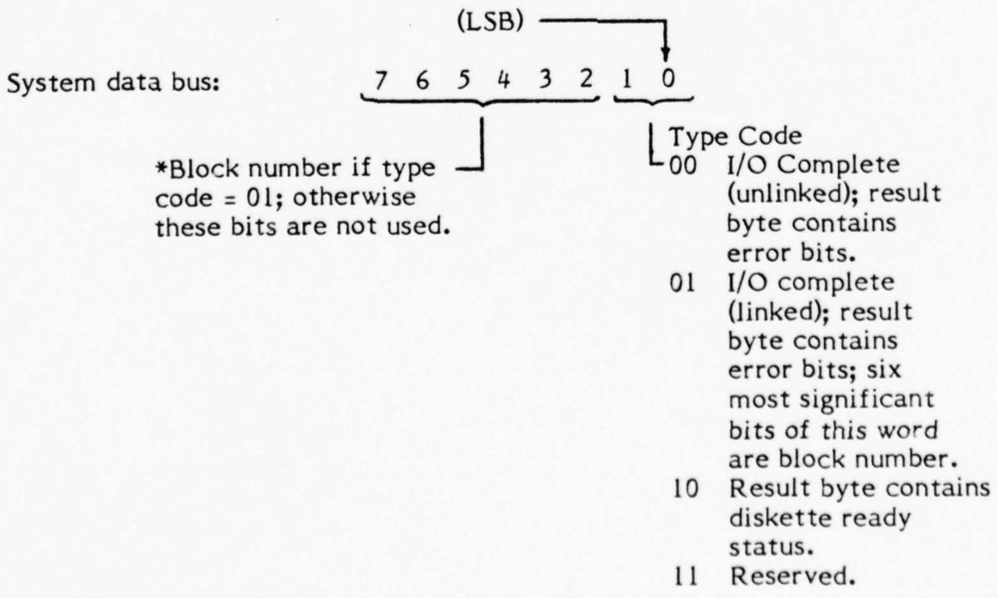
These indications allow the operating system to monitor the operation of the Diskette Channel.

System address bus: BASE + Ø


(LSB)
System data bus:
7 6 5 4 3 2 1 0
NOT USED
logical 1 = controller present
logical 0=controller not present
logical 1=interrupt pending
logical 0=no interrupt pending
logical 1=drive 1 ready
logical 0=drive 1 not ready
logical 1=drive 0 ready
logical 0-drive 0 not ready

Read Result Type (input). This input channel command causes the Diskette Channel to return eight bits of information to the CPU. The two least significant bits specify one of four different types of result byte (see next paragraph) associated with diskette operations. The remaining six bits are interpreted according to the code presented in the two least significant bits.

System address bus: BASE + 1

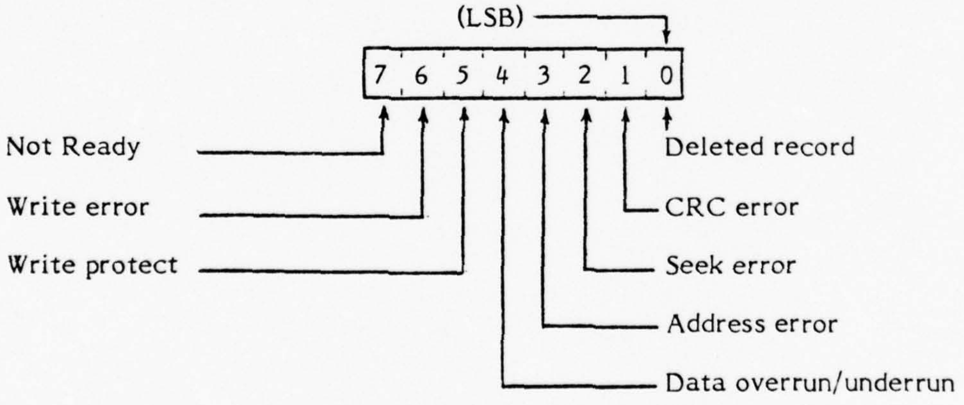

(LSB)
System data bus:
7 6 5 4 3 2 1 0
*Block number if type code = 01; otherwise these bits are not used.
Type Code
00 I/O Complete (unlinked); result byte contains error bits.
01 I/O complete (linked); result byte contains error bits; six most significant bits of this word are block number.
10 Result byte contains diskette ready status.
11 Reserved.

*NOTE: A block number is a 6-bit binary number that uniquely identifies an IOPB. The block number is returned in the result type word to identify the IOPB associated with the current interrupt. This scheme is only required when several IOPB's are linked together to perform multiple operations, because there is no uncertainty about the origin of an interrupt request when only a single IOPB exists.

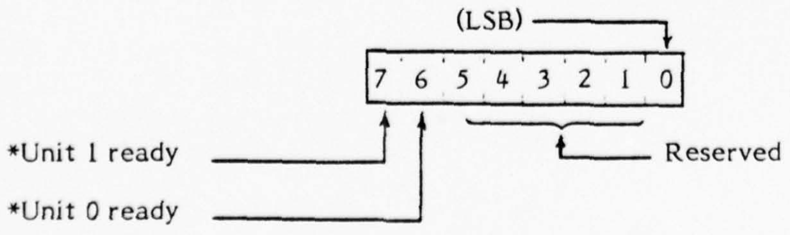
Read Result Byte (input). This input channel command causes the Diskette Channel to return eight bits of information to the CPU. The interpretation of these bits is dependent upon the type code returned in the result type word (see previous paragraph). The "read result byte" channel command should only be executed after a "read result type" command has been executed.

System address bus: BASE + 3

System data bus: If the type code in the result type word = 00 or 01, the result byte, input on the data bus, will contain error bits (see for error explanations) and will be formatted as follows:



If the type code = 10, the controller has detected a change in the ready status of a drive and contents of the result byte will indicate the current ready status of the diskette drives:



*NOTE: A logical 1 means that the drive is currently ready; a logical 0 means the drive is not ready. It is the responsiblity of the host system software to maintain appropriate table to track these status changes. There is one instance in which a drive can appear "not ready" to the host system, when in fact it is ready. For example, assume that while drive 0 is selected, drive 1 just goes not ready then returns to the ready state (perhaps the diskette platter was changed). When the drive 0 operation is completed, the diskette controller will return two consecutive status change interrupts, the first showing drive 1 not ready, the second showing drive 1 ready. The first interrupt, indicating drive 1 to be not ready, is returned even though the drive is now actually ready because it is important that the operator know that the ready status of the drive changed while the other drive was selected. For instance, this would protect against inadvertently accessing an "unknown" disk, if the drive went not ready then ready again because someone changed disk platters.

Diskette Operations

The Diskette System is capable of performing seven different operations: recalibrate, seek, format track, write data (with data marks), read data, and verify CRC. To initiate any diskette operation (or group of linked operations), the CPU will output both bytes of the 16-bit memory address that points to the first byte of an I/O Parameter Block (IOPB). The second byte in the IOPB specifies one of the seven diskette operations. After the Diskette System receives the upper byte of the 16-bit memory address, it accesses the IOPB to determine the operation to be performed and to acquire the various parameters that are necessary for execution of the diskette instruction. The Diskette System will perform the specified operation, then set its interrupt flip-flop. If several IOPB's are linked together, the Diskette System will perform all of the specified operations (each IOPB specifies one diskette operation) before interrupting the CPU with a request for service.

NOTE: The Diskette Channel automatically unloads the read/write head after a fixed length of time following a diskette operation or group of linked operations. This feature is meant to reduce head wear. The feature is implemented by counting index pulses after a "read result byte" channel command is executed. When the specified count is achieved, the head is unloaded, and the count is re-initialized. At present, the count is set for 6; that is, the head will remain loaded for at least five complete revolutions following each diskette operation or group of linked diskette operations.

The seven diskette operations are defined in the following paragraphs:

Recalibrate. This operation causes the head of the selected diskette unit to be moved over track 00. The diskette drive's track 0 sensor is sampled to determine successful completion of this operation. This is often the first instruction executed after a diskette is loaded, or when a seek error occurs.

Seek. This operation causes the head of the selected diskette unit to be moved over the track specified in byte 4 of the IOPB. The Diskette Channel will verify the head position by reading the track address from the diskette platter before completing the operation. If at the completion of the head movement, the head is not over the expected track a "seek error" will be indicated.

Format Track. This operation initializes the track specified in byte 4 of the IOPB, by writing all address marks, gaps, address fields and data fields, as shown in Figure 4-2. The various address marks and fields are defined in Appendix I.

The method of assigning logical sector addresses, which are written into the sector address fields, is specified by bit 6 of the first IOPB byte (the channel word). If this bit is equal to logical 0 the sequence of logical sector addresses will match the physical sequence on the diskette (i.e., sector address "01" is written into the first physical sector after the index mark, sector address "02" is written into the second physical sector, and so on). In addition, the data byte stored in the memory location specified by

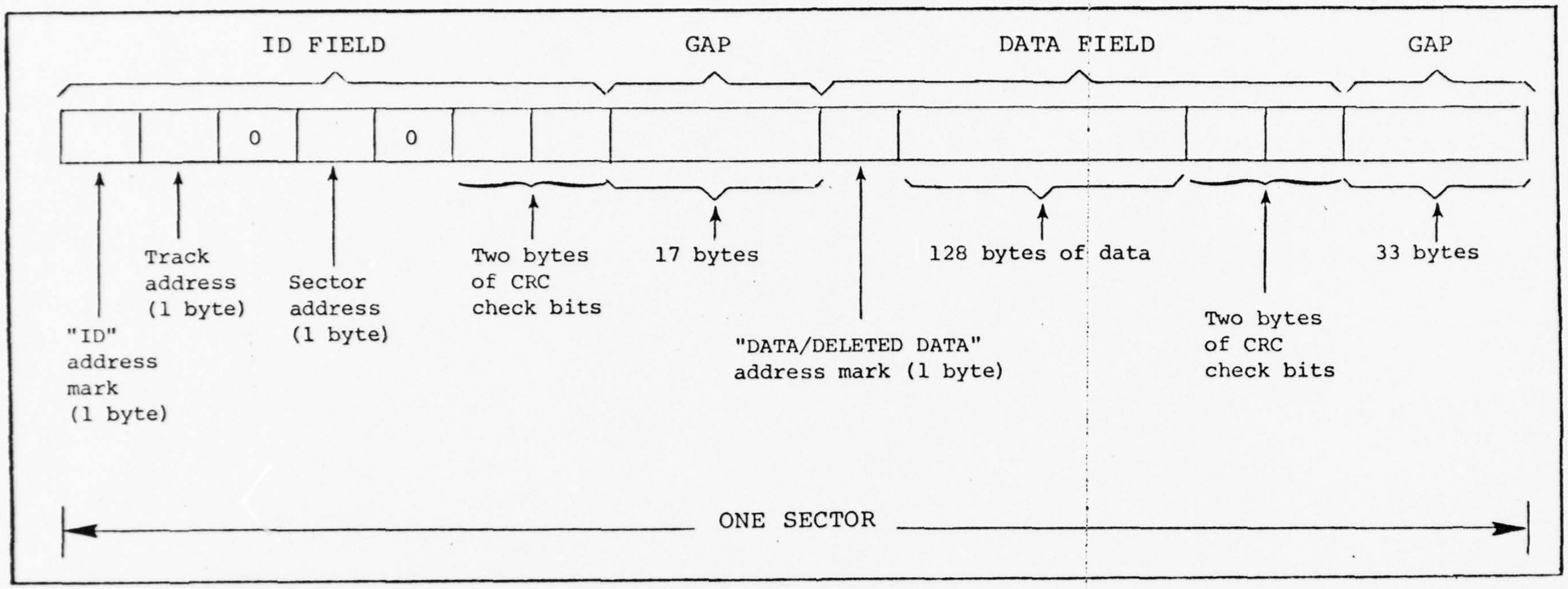

ID FIELD
GAP
DATA FIELD
GAP
0
0
"ID" address mark (1 byte)
Track address (1 byte)
Sector address (1 byte)
Two bytes of CRC check bits
17 bytes
"DATA/DELETED DATA" address mark (1 byte)
128 bytes of data
Two bytes of CRC check bits
33 bytes
ONE SECTOR

Fig. 4-2. Sector Format

the 16–bit buffer address contained in bytes 6 and 7 of the IOPB will be written into the 128-byte locations of each sector's data field. No other data bytes need to be stored in this buffer.

If on the other hand, the sequence of logical addresses being assigned to the sectors is "random" (that is, do not match the physical sequence of sectors), bit 6 of the channel word will be equal to logical 1, and 52 bytes (26 pairs) of data will be stored in memory beginning at the 16–bit buffer address contained in bytes 6 and 7 of the IOPB. Each of the 26 pairs of data bytes will specify the logical sector address to be written into the sector address field of the corresponding physical sector, and the data character which will be written (128 times) into the data field portion of that sector. For example, if the first four bytes of the buffer are:

Byte	Contents (hex)
1	01
2	FF
3	0E
4	00
.	.
.	.

Then, sector address "01" will be written into the sector address field of the first physical sector after the index mark, and "FF_{16}" (all ones) will be written into each of the 128 byte locations in the data field portion of this sector. The sector address "$0E_{16}$" (14_{10}) will be written into the sector address field of the second physical sector (i.e., the sector which is physically next to the first sector), and "00_{16}" (all zeros) will be written into each of the 128 byte locations in the data field portion of this sector. And so on, until a logical sector address has been written into the sector address field of each of the 26 physical sectors on the track, and a data byte is written into each of the 128 byte locations in the data field portion of each of the 26 sectors.

The firmware implementation of the format command is such that in order to format track n (n≠0), track n-1 must already be formatted (i.e., already have readable address information written into it). Track 0 can always be formatted, even if no valid address information is written on the disk. During formatting, a "data mark" (i.e., a character which has a clock pattern equal to $C7_{16}$ and a data pattern equal to FB_{16}; see Figure 4-3) is written into the "data/deleted data address mark" character position of each sector (i.e., the character position immediately preceding the 128 byte data field).

If, when the format track operation is initiated, the head is not already positioned over the track specified in byte 4 of the IOPB, the format track instruction will cause the head to move (seek) to the proper track before the actual formatting begins.

Write Data. This operation transfers N x 128 bytes of contiguous data from memory to the diskette. N represents the number of sectors to be written. N is specified by the contents of byte 3 of the IOPB. The 16-bit buffer address stored in bytes 6 and 7 of the IOPB specifies the memory location containing the first data byte to be transferred. The contents of bytes 4 and 5 of the IOPB (track and sector addresses, respectively) specify the logical address of the first sector to be written into.

Each 128 byte data field will be preceded by a "data" address mark (see Figure 4-3) that is used for synchronization. Two bytes (16 bits) of CRC check bits will be generated and written after each data field; the CRC bytes are generated from the address mark, as well as the 128 data bytes.

A multi-sector operation (i.e., $N \geq 2$) may begin at any sector, but must not go beyond the last logical sector on a track (sector 26).

If the head is not already positioned over the track specified in byte 4 of the IOPB, the write data instruction will cause the head to move (seek) to the proper track before the actual writing begins.

Write "Deleted" Data. This operation is identical to the WRITE DATA operation, described above, except that each 128 byte data field is preceded by a "deleted data" address mark, shown in Figure 4-4.

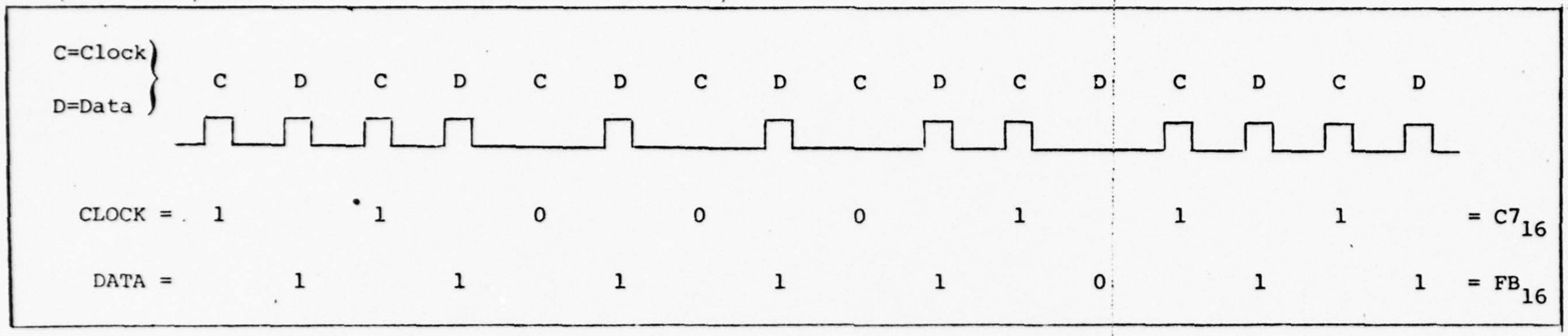

C=Clock
D=Data
C D C D C D C D C D C D C D C D
CLOCK = 1 1 0 0 0 1 1 1 = $C7_{16}$
DATA = 1 1 1 1 1 0 1 1 = FB_{16}

Fig. 4-3. Data Address Mark

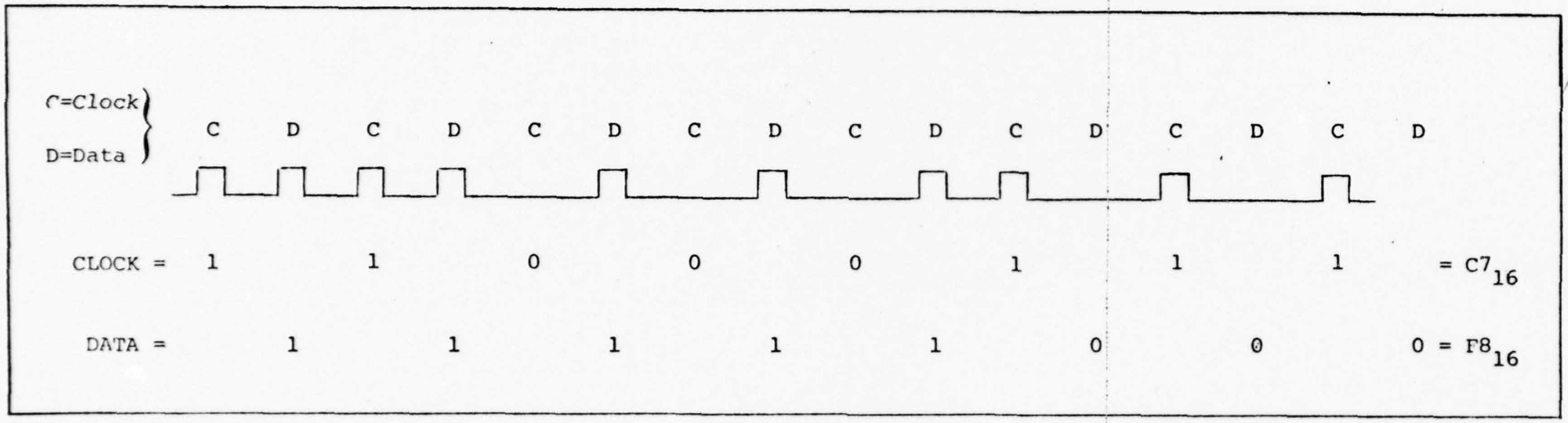

C=Clock
D=Data
C D C D C D C D C D C D C D C D
CLOCK = 1 1 0 0 0 1 1 1 = $C7_{16}$
DATA = 1 1 1 1 1 0 0 0 = $F8_{16}$

Fig. 4-4. Deleted Data Address Mark

Read Data. This operation transfers N sectors of data (128 bytes per sector) from diskette to memory. N is specified by the contents of byte 3 of the IOPB. The contents of bytes 4 and 5 of the IOPB (track and sector addresses, respectively) specify the logical address of the first sector to be read. The 16-bit buffer address stored in bytes 6 and 7 of the IOPB specifies the memory location into which the first data byte will be written.

Two bytes of CRC check bits will be generated as each sector is being read. When the "data" address mark and all 128 data bytes of a sector have been read, the generated CRC bits are compared with the 16 CRC bits previously written. If there is a mismatch, a CRC error is indicated.

A multi-sector operation (i.e., $N \geq 2$) may begin at any sector, but must not go beyond the last logical sector on a track sector 26).

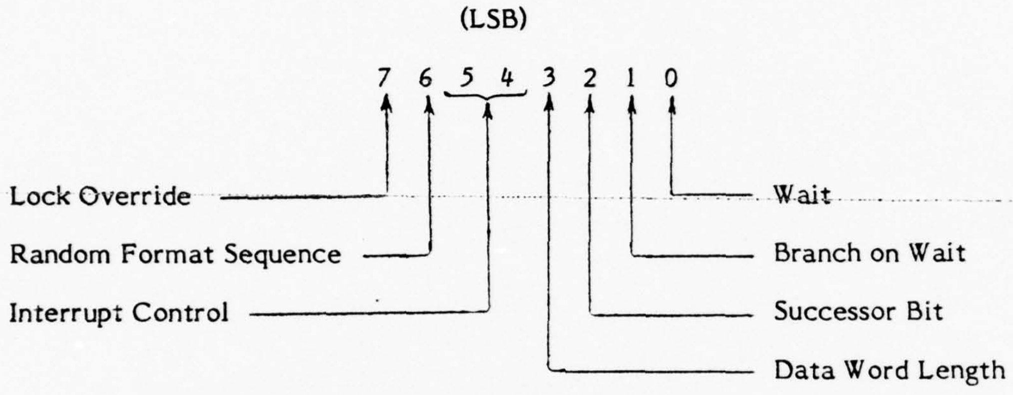
If the head is not already positioned over the track specified in byte 4 of the IOPB, the read data instruction will cause the head to move (seek) to the proper track before the actual data reading begins.

Verify CRC. This operation is identical to the READ DATA operation, described above, except that no data is transferred to memory.

I/O Parameter Block

The CPU initiates a diskette operation, or group of linked operations, by outputting a 16-bit address that points to the beginning (the channel word) of the first (or only) I/O Parameter Block (IOPB) in system memory. The Diskette Channel then accesses the IOPB. An IOPB specifies one of the seven diskette operations and provides all of the parameters required for the completion of that operation. An IOPB consists of ten bytes, as shown in Figure 4-5.

Byte 1: Channel Word. This byte contains channel control information to be used by the Diskette System. Bit assignments in this byte are as follows:



The "random format sequence" bit (6) specifies the method of assigning logical sector addresses when formatting a track. If this bit is reset (logical 0), sector addresses are assigned in sequential order. If this bit is set (logical 1), sector addresses are assigned in random order according to the pattern listed in the 52 byte memory buffer, which begins at the location addressed by the contents of IOPB bytes 6 and 7.

The "interrupt control" bits (4 and 5) enable or disable Diskette Channel interrupts according to the scheme shown in Table VI.

BYTE	
*1	Channel Word
2	Diskette Instruction
3	Number of Records
4	Track Address
5	Sector Address
6	Buffer Address (Lower)
7	Buffer Address (Upper)
8	Block Number
9	Next IOPB Address (Lower)
10	Next IOPB Address (Upper)

*The 16-bit address output to the Diskette System by the two "Write MA" channel commands points to the first byte of an IOPB.

Fig. 4-5. I/O Parameter Block (IOPB) Format

TABLE VI

Interrupt Control Bits

BIT: 5	4	FUNCTION
0	0	I/O complete interrupt request to be issued (a) upon completion of an unchained diskette operation, (b) upon completion of the last operation in a chain of linked operations, or (c) upon detection of an error in any intermediate operation in a chain of linked operations.
0	1	All I/O complete interrupts are disabled.
1	0	I/O complete interrupt to be issued after current operation even though it is an intermediate link in a chain.
1	1	Illegal code

NOTE: The interrupt control bits do not affect interrupt requests which are issued as the result of a change in diskette ready status.

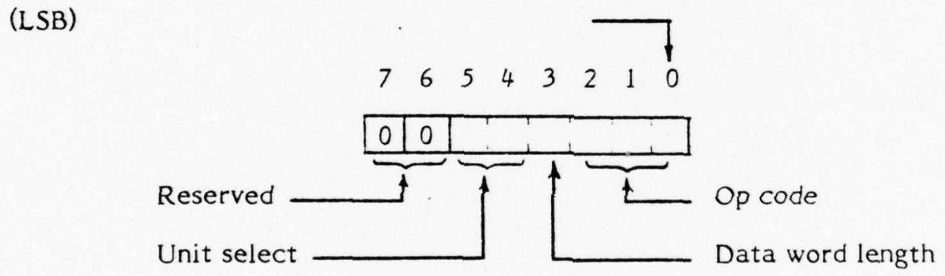
The "data word length" bit (3) must be reset (logical 0) when the Diskette Channel is being used with 8-bit systems, or set (logical 1) when being used with 16-bit systems. This bit must be logical 0 when being used with the MTC system (a 32-bit system) because the interface will be designed to transfer 8-bit bytes of data as noted in the specifications.

The "successor" bit (2) will be reset (logical 0) if the current IOPB is the last (or only) IOPB to be executed. This bit will be set (logical 1) if at least one more IOPB is to follow. If this bit is reset, an I/O complete interrupt request will be issued when the operation specified by the current IOPB is completed. If there is a successor IOPB, that IOPB will begin at the memory location specified by bytes 9 and 10 of the current IOPB.

The "branch on wait" (1) and "wait" (0) bits are interrelated. If the "wait" bit is reset (logical 0), the Diskette Channel immediately performs the diskette operation specified by the current IOPB. If the "wait" bit is set (logical 1), however, the Diskette Channel examines the "branch on wait" bit. If "branch on wait" is set (logical 1) and "wait" is set, the Diskette Channel performs an unconditional branch to the memory location specified by the 16-bit address in bytes 9 and 10 of the current IOPB; the next IOPB to be executed should reside at this memory location. If "branch on wait" is reset (logical 0) but "wait" is set, the Diskette Channel will idle for 10 msec., then examine the "wait" bit again, remaining in this loop until "wait" is reset. This gives the programmer greater flexibility in establishing a sequence of operations to be performed by the Diskette Channel. Note that "branch on wait" MUST be reset (logical 0) if "wait" is reset (logical 0). The Diskette Channel sets the "wait" bit after completing the operation specified in the IOPB (unless the "lock override" bit is set). This protects again inadvertent execution of an uninitialized IOPB.

The "lock override" bit (7) when set, specifies that the "wait" bit is not to be set upon completion of the operation specified in the IOPB. This prevents the IOPB from being overwritten by the controller which is a useful feature during system debugging.

Byte 2: Diskette Instruction. This byte specifies the diskette operation to be performed and identifies the diskette unit to be used:



The "unit select" bits (4-5) specify drive #0 when reset (logical 00_2), or drive #1 when set (logical 11_2). The "data word length" must contain the same value as the corresponding bit in the channel word (byte 1). The "op code" bits (0-2) specify one of the seven diskette operations:

BIT: 3	2	1	OPERATION
0	0	0	No operation
0	0	1	Seek
0	1	0	Format track
0	1	1	Recalibrate
1	0	0	Read data
1	0	1	Verify CRC
1	1	0	Write data
1	1	1	Write "Deleted" Data

Byte 3: Number of Records. This binary number specifies the number of sectors to be transferred. Multi-sector operations are allowed, but they must not go beyond the last sector on a track (sector 26); that is, an address error will be indicated if (starting sector address) + (number of records) 26_{10}. Therefore, the maximum block transfer is 26 sectors (from sector 1 to sector 26).

Byte 4: Track Address. This binary number identifies the track. Acceptable values are 0 to $4C_{16}$ (76_{10}), inclusive.

Byte 5: Sector Address. Bits 4 through 0 of this byte contain a binary number which specifies the first sector to be accessed during transfer operations. Acceptable values are 1 to $1A_{16}$ (26_{10}), inclusive. Bit 5 of this word MUST correspond to bit 5 (select bit) of the diskette instruction word (byte 2). Bits 6 and 7 are not used.

Byte 6: Buffer Address (lower). This byte contains the eight least significant bits of the 16-bit buffer memory address.

Byte 7: Buffer Address (upper). This byte contains the eight most significant bits of the 16-bit buffer memory address. Bytes 6 and 7 together contain the 16-bit address of the first word of the buffer in system memory. During read data operations, the data from the diskette is transferred to the buffer. During write operations, data from the buffer is written to diskette. During format track operations, the address assignment pattern and/or the data field "format characters" are stored in the buffer.

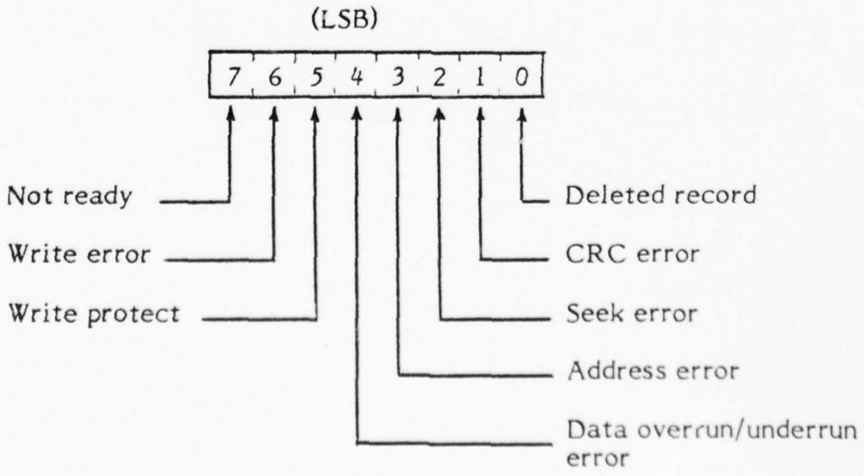
Byte 8: Block Number. This byte contains a 6-bit (right-justified) binary number that uniquely identifies the current IOPB. The block number allows the CPU to associated an I/O complete interrupt request from an intermediate link in a chain of IOPB's with the IOPB which actually caused the interrupt. The block number need only be initialized for linked IOPB's, since there can be no uncertainty when only a single IOPB exists.

Byte 9: Next IOPB Address (lower). This byte contains the eight least significant bits of the 16-bit memory address that points to the beginning of the next IOPB in the chain.

Byte 10: Next IOPB Address (upper). This byte contains the eight most significant bits of that 16-bit memory address that points to the first byte of the next IOPB in a chain. The Diskette System will access the next IOPB after the current operation if the "successor" bit (byte 1) in the current IOPB is set, or immediately if both the "wait" and "branch on wait" bits (byte 1) in the current IOPB are set.

Error Indications

If the CPU executes a "read result byte" channel command (in response to a "read result type" channel command which returned a code of 00 or 01), the Diskette Channel will return the following result word on the system data bus:



The bits are defined as follows:

Not Ready. This bit (7) indicates that the selected unit was not ready or that the selected unit changed to a not ready state during an operation.

Write Error. This bit (6) indicates that, during a write operation, a condition existed which precluded data integrity. This error is detected by the drive and monitored by the Diskette Channel controller. An example of a condition that could cause this error is an attempt to write through an unloaded head.

Write Protect. This bit (5) indicates that the selected drive contains a diskette platter which is in the "read only" mode. This condition is checked on format track, write data (with data address marks) and write data (with deleted data address marks) operations.

Data Overrun/Underrun Error. This bit (4) indicates that the Diskette System controller was not able to service a byte transfer request from the drive before the next request occurred. The data byte is "lost".

Address Error. This bit (3) indicates that the disk address received from the CPU is invalid; that is:

- track address $>76_{10}$,
- sector address = 00,
- sector addres $>26_{10}$, or
- sector address + number of records $>26_{10}$

Seek Error. This bit (2) indicates that, at the completion of a head movement sequence, the head is not positioned over the expected track. This bit indicates the Diskette System controller and/or drive are malfunctioning, and a recalibrate diskette operation should be performed. Because all of the diskette operations may implicitly cause the head to move, a seek error can occur during any diskette operation.

CRC Error. This bit (1) indicates that the two CRC characters generated during a read data or verify CRC operation were not the same as the two CRC characters appended to the data field when it was written on diskette.

Deleted Record. This bit (0) indicates that a sector addressed during a read data or verify CRC operation was preceded by a deleted data address mark.

Four other error conditions are indicated when more than one error bit is true:

1. ID CRC Error. If the address error (3) and CRC error (1) bits are true, it indicates that the CRC characters generated during the reading of an ID field were not the same as the CRC characters appended to the field when it was written by a format track operation.

2. SYNC Error. If the deleted record (0) and CRC error (1) bits are true, it indicates that an unexpected address mark pattern was encountered while trying to establish data synchronization. This usually indicates that the ID field is garbled.

3. NO ADDRESS MARK. If the address error (3), seek error (2) and CRC error (1) bits are true, it indicates that no address mark was encountered for a full revolution of the diskette. This usually indicates that the track has not been formatted.

4. DATA MARK Error. If the address error (3), seek error (2), CRC error (1), and deleted record (0) bits are true, it indicates that the data field of a particular sector was not preceded by either a data mark or a deleted data mark.

The Channel Board

For description purposes, the circuitry on the Channel Board can be divided into six functional blocks:

(1) Channel Command Block

(2) Micro Control Unit (MCU) Block

(3) Microprogram Memory Block

(4) Central Processing Element (CPE) Block

(5) Data/Clock Shift Register (SR) Block

(6) Data Flow Control Block

as shown in Figure 4-6.

Channel Command Block. The channel command block is responsible for recognizing and decoding channel commands being executed by a CPU. When the channel command block recognizes the switch-selectable BASE address of the Diskette Operating System on the interface system address bus, it decodes the three least significant address bits (ADRØ/ - ADR2) to determine which of the seven channel commands is being executed. The three address bits are also latched and made available to the MCU block, which is ultimately responsible for controlling the diskette controller's response to a channel command. The channel command block also includes the interrupt latch which stores the fact that an interrupt request has been issued to the CPU by the microprogram.

Micro Control Unit Block. The micro control unit (MCU) block accepts and decodes the three address bits from the channel command block (ADRØ/ - ADR2/) specifying a channel command or the three least significant data outputs from the CPE block (DØ-D2) specifying one of the seven diskette operations. The two groups of three bits select one of the

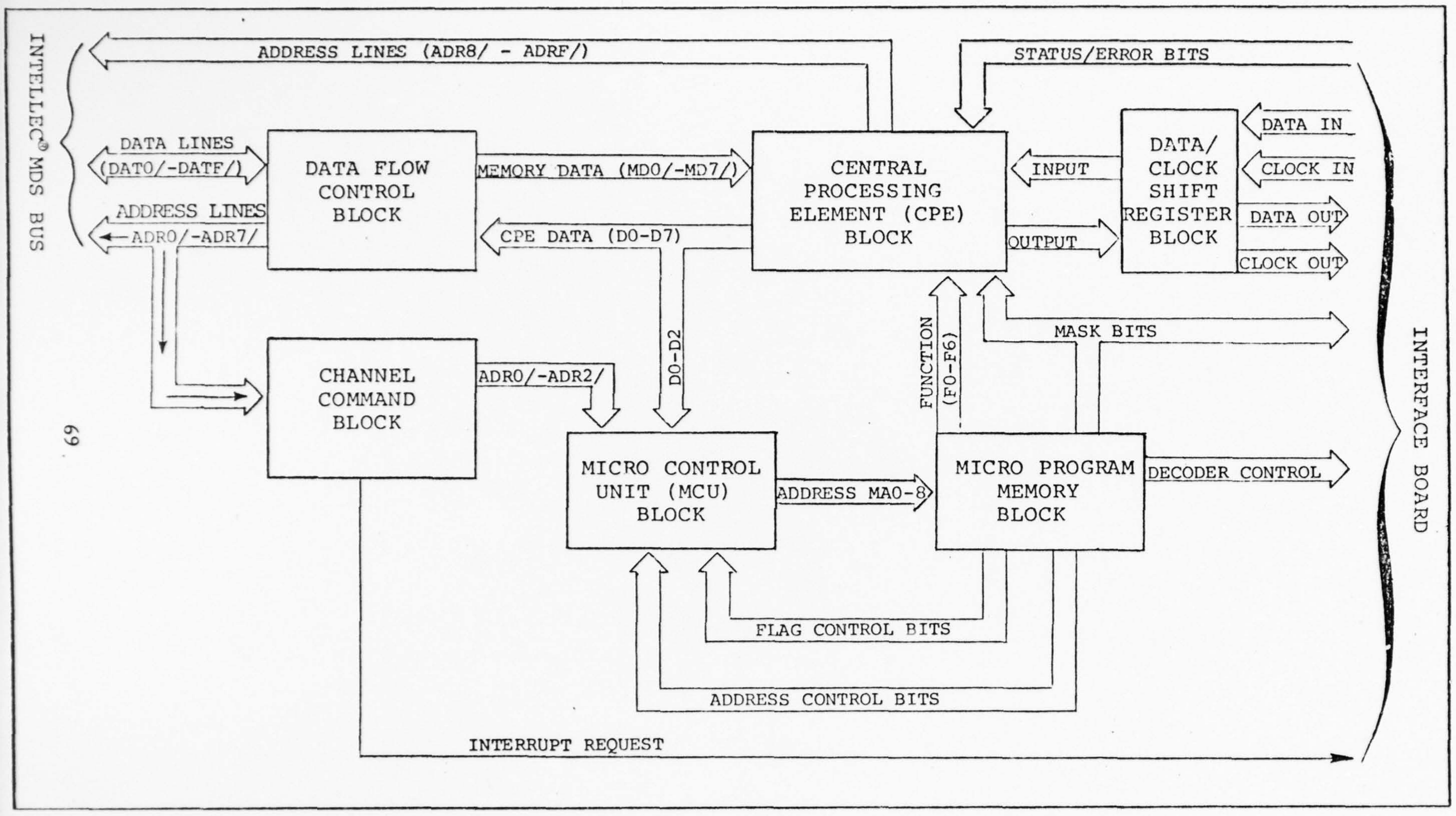


Fig. 4-6. Channel Board - Functional Block Diagram

ten routines which implement the channel commands and I/O operations. Having determined the microprogram routine to be executed, the MCU block then generates and outputs the appropriate nine-bit memory address from the micro-program memory. The MCU continuously examines the two flag control lines and the seven address control lines (AC0 - AC6) from the micro-program memory block to determine the address of the next micro-instruction to be fetched and executed.

Microprogram Memory Block. The microprogram memory block, as its name implies, stores the microprogram. The microprogram memory is organized into 512 words of 32 bits each. The nine address bits from the MCU block determine which 32-bit microinstruction will be output from the microprogram memory. Nine bits of the microinstruction (the address control and flag control bits) are applied to the MCU block, as mentioned above, while the seven function bits (F0-F6) are applied to the CPE block and specify the operation to be performed by the processing elements. The other 16 bits of the microinstruction words perform a variety of control functions.

CPE Block. The CPE block includes four Intel 3002 Central Processing Elements, which form an 8-bit processor. The CPE block receives data from the data flow control block and the data/clock shift register block and receives status information from the Interface Board. The CPE can operate on these various types of input data under the direction of the function and mask control bits from the microprogram memory. The results of these arithmetic/logical operations can then be output onto the eight most significant system address lines (ADR8/ - ADRF/) or the eight CPE data lines D0 - D7). D∅ - D7 are, in turn, made available to the MCU block, the data/clock shift register block and the data flow control block.

Data/Clock Shift Register Block. The data/clock shift register block includes the shift registers that accept the serial data bits and the serial clock bits and input them, in parallel, to the CPE block during read operations. During write operations, the data and clock bytes are (parallel) loaded into the shift registers from the CPE block and shfted out (serially) to the Interface Board.

Data Flow Control Block. The data flow control block routes data from the CPE data lines (D0 - D7) to the eight least MDS system address lines (ADØ/ - ADR7/) or to either the lower or upper eight lines of the system data bus (DATA0/ - DATA7/ or DATA8/ - DATAF/). This block also routes data from either half of the data bus onto the memory data input lines (MD0/ - MD7/) that feed the CPE block.

The Interface Board

The circuitry on the Interface Board can be divided into five functional blocks as follows: (See Fig. 4-7.)

- Disk drive control block
- Serial data/clock synchronization block
- Write clock generator block
- Cyclic Redundancy Check (CRC) block
- Bus control block

Disk Drive Control Block. The disk drive control block provides the unit selection/head loading (SELn/) signal, the direction indicator (DIR/) and the step pulse (STEP/) that moves the read/write head on the selected unit one track in the specified direction. The disk drive control block also monitors the READY status, the INDEX indicator and the TRACK0 indicator from the two drives.

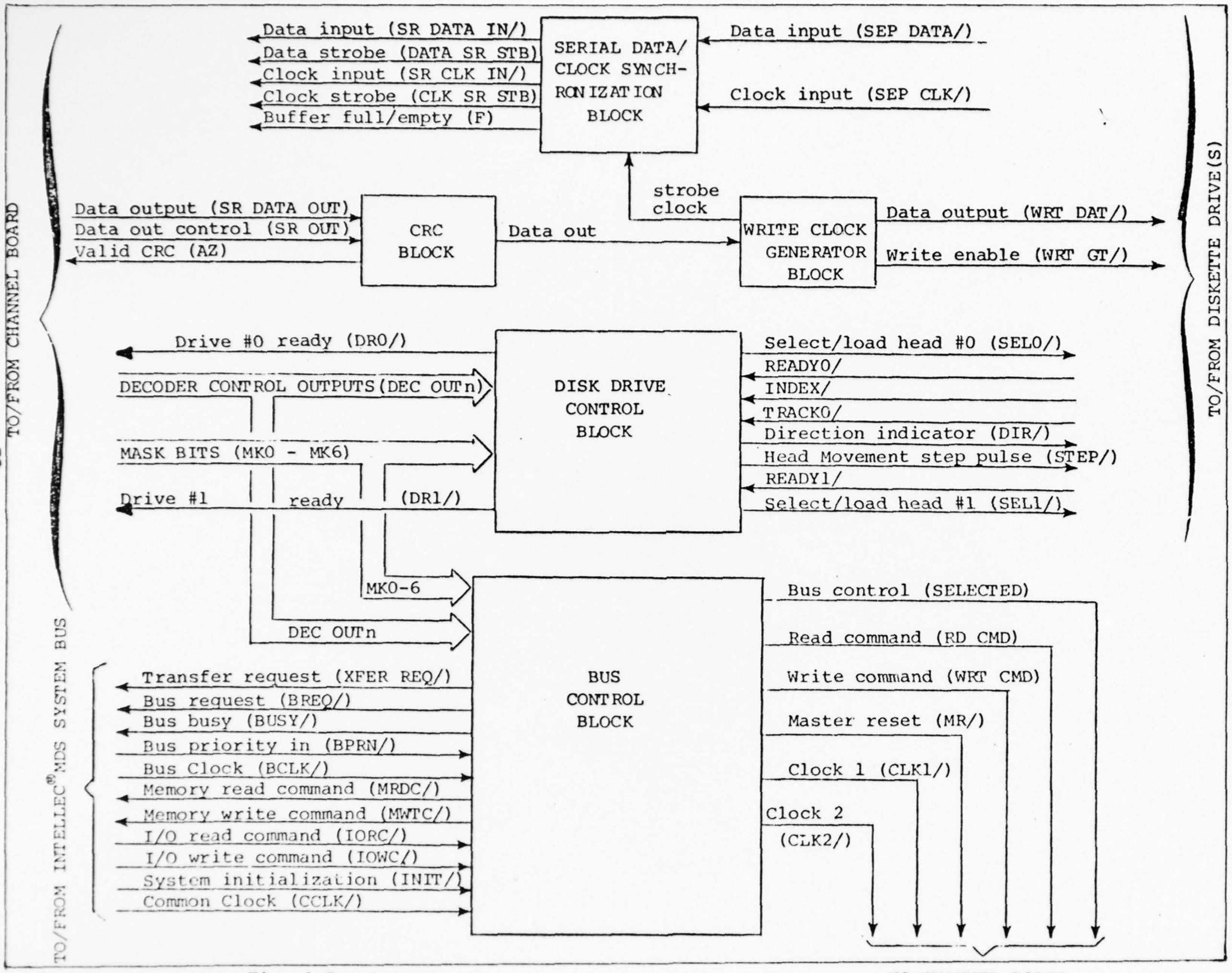


Fig. 4-7. Interface Board - Functional Block Diagram

Serial Data/Clock Synchronization Block. The serial data/clock synchronization block receives the separated data and clock bits, and examines the bit patterns looking for specific patterns which indicate an address mark. Address marks precede address and data fields and are used to synchronize the controller with the drive. The synchronization block then generates data and clock strobes (DATA SR STB and CLK SR STB) which shift the data and clock bits into the shift registers on the Channel Board. The synchronization block also includes a bit counter that determines when a byte (8-bits) has been shifted to/from the selected drive.

Write Clock Generator Block. The write clock generator block provides timing references for the writing of data and clock bits. Data and clock bits are both output to the drive via the WRT DAT/ line.

CRC Block. The CRC block generates two CRC characters (16 bits) which are suspended to the end of each address field and data field during format and write data operations. During read operations, the CRC block generates two CRC characters for each data field (includes address mark and 128 bytes of data) read, then compares these with the two CRC characters that were appended to the data field, to verify the validity of the data.

Bus Control Block. The bus control block provides the interface with the system bus. The bus control block requests and maintains master control of the system bus, and generates the memory read (MRDC/) and memory write (MWTC/) commands that allow the diskette controller to access system memory. In addition, the bus control block acknowledges (XACK/) the I/O read (IORC/) or I/O write (IOWC/) command that is issued when the CPU in the system executes a channel command to the diskette controller.

Base Address Selection

The user must assign a base address to the Diskette Channel. The base address is defined by the five most significant bits of the eight-bit I/O port address. The three least significant bits, then, can be be used to differentiate between eight input or eight output channel commands. When the CPU accesses the Diskette Channel by executing an I/O instruction, the base address (BASE) is used to select the Diskette Channel, while the three low-order address bits select one of the channnel commands, as described earlier in this chapter.

A base address is assigned by opening or closing the five most significant switch positions of the S1 switch (S1-4,5,6,7,8) on the Channel Board (see sheet 1 of the Channel Board schematic in Appendix D). When a switch position is closed (on) (tied to ground) it represents the assignment of a logical 0 address bit. When a position is open (off) (+5V), it represents a logical 1 selection.

The following sketch represents a base address selection of 78_{16}.

8 TOP OF BOARD

7

6

5

4

3

2

1

ON

For this application, the base address selection has been arbitrarily set to 78_{16}.

Summary

This chapter has given the reader a detailed look at Intell's DOS diskette controller and the interface signals necessary for the DOS to function properly. The functional operations of the interface and channel boards of the diskette controller were explained. The channel commands, diskette instructions, I/O parameter block, and error conditions were each fully discussed as these areas constitute the broad spectrum of understanding essential to the development of the interface design. The reader has been introduced to the systems which are being interfaced and should now be ready to understand the symbolic design which follows in the next chapter.

V. SYMBOLIC DESIGN

This chapter discusses the symbolic design derivation and specifies it in terms of register transfer language. The reader should reference Figure 5-3 throughout the reading of this chapter.

Symbolic Design Approach

The functional decomposition for the logical design of the required interface for this particular project is shown in Figure 5-1. A quick analysis of this chart will reveal that it contains the essential steps required of any interface design approach. Namely, the CPU must be capable of telling the peripheral device to do something through I/O instructions; the peripheral device must then be able to interpret the instruction and take the necessary action; and finally, the peripheral device, when it has completed the required actions, must interrupt or in some way inform the CPU that the action has been completed. The CPU can then continue its processing. The reader should note, at the outset, that all output to or input from the disk to the ACU will be accomplished under program control.

The chart of Figure 5-1 was constructed after the MTC and DOS were both carefully studied. From this design chart comes the essential characteristics and hardware features that must be accomplished by the interface. It is the starting point for the design. The entire design is generated around the requirements noted in each of the blocks. After gaining the necessary understanding and knowledge of each of block requirements, the actual logical design of the interface can begin.

Before taking the first step of laying out the bus structures, the decision to incorporate direct-memory access (DMA) will be discussed.

A major design decision in the development of an interface is to determine whether or not the interface will allow peripheral devices to perform direct-memory accesses. Hardware logic which interfaces a parallel data bus directly with the digital-processor memory data register, which allows requests, and which grants 1-cycle pauses in processor operation for direct transfer of data to or from memory, is referred to as direct-memory access (DMA). There are several reasons for designing the interface to do this.

First, it allows the system to transfer data blocks at very high rates (250K words/sec) without elaborate I/O programming. In this design, as with most DMS facilities, only a few I/O instructions are needed to initialize or reinitialize transfers. Thus, it eliminates tedious step-by-step programming which is a second advantage. Another advantage for using DMA, due to its fast data transfer, is it's suitable for servicing peripherals with high data rates, such as magnetic disks, and drums. Finally, if there are many devices to be serviced, minimal program overhead is attained which is extremely valuable in many applications, for example, in flight simulators where multiple data blocks must be transferred in real time. For these reasons, DMA is being designed into the interface.

It is also true that DMA makes the design more sophisticated and complex, but the additional capability is well worth the time and effort spent. The reader is now ready to look at the first step in the symbolic design phase.

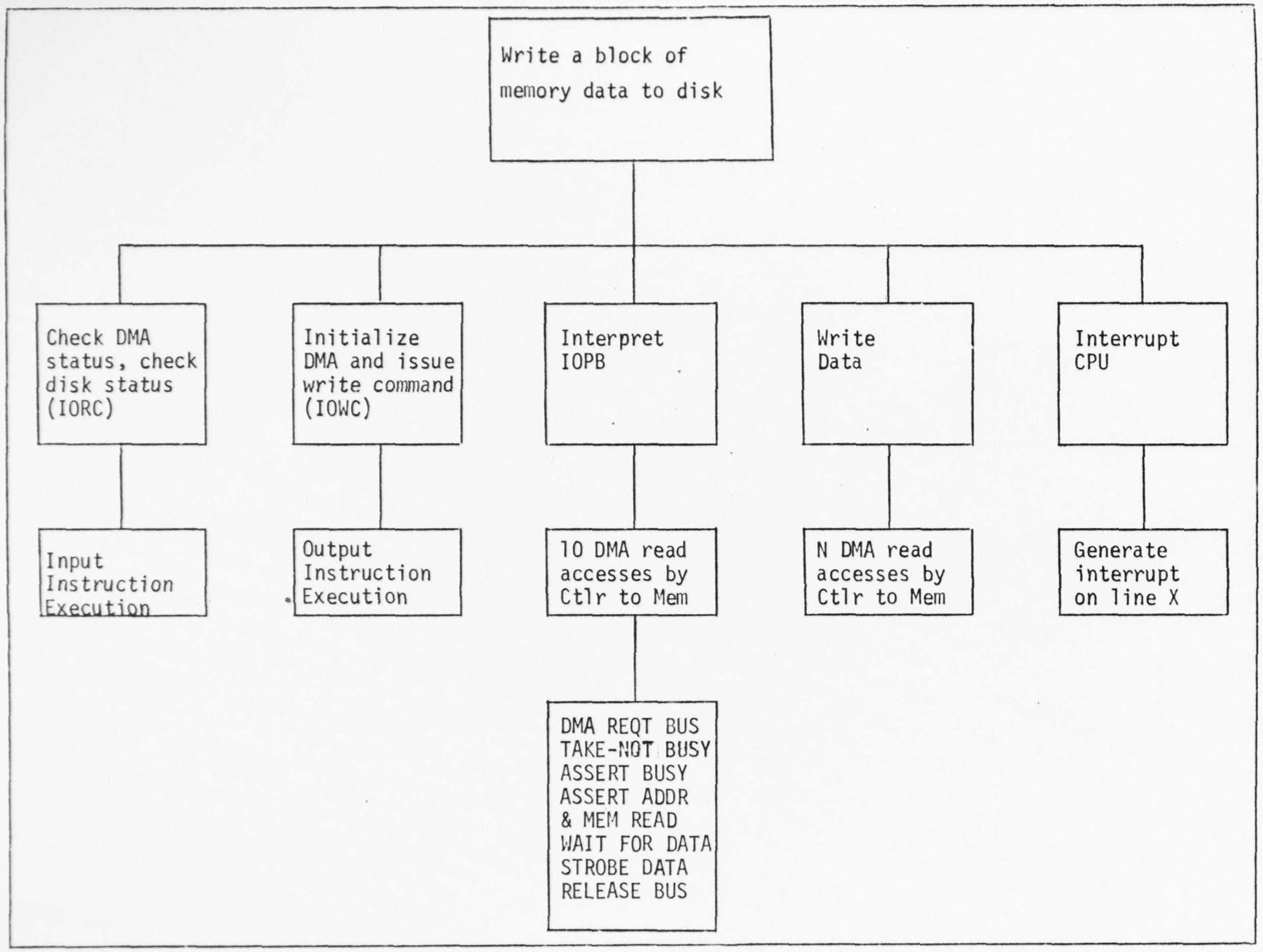

Write a block of memory data to disk
Check DMA status, check disk status (IORC)
Initialize DMA and issue write command (IOWC)
Interpret IOPB
Write Data
Interrupt CPU
Input Instruction Execution
Output Instruction Execution
10 DMA read accesses by Ctlr to Mem
N DMA read accesses by Ctlr to Mem
Generate interrupt on line X
DMA REQT BUS
TAKE-NOT BUSY
ASSERT BUSY
ASSERT ADDR
& MEM READ
WAIT FOR DATA
STROBE DATA
RELEASE BUS

Fig. 5-1. Functional Interface Decomposition Chart

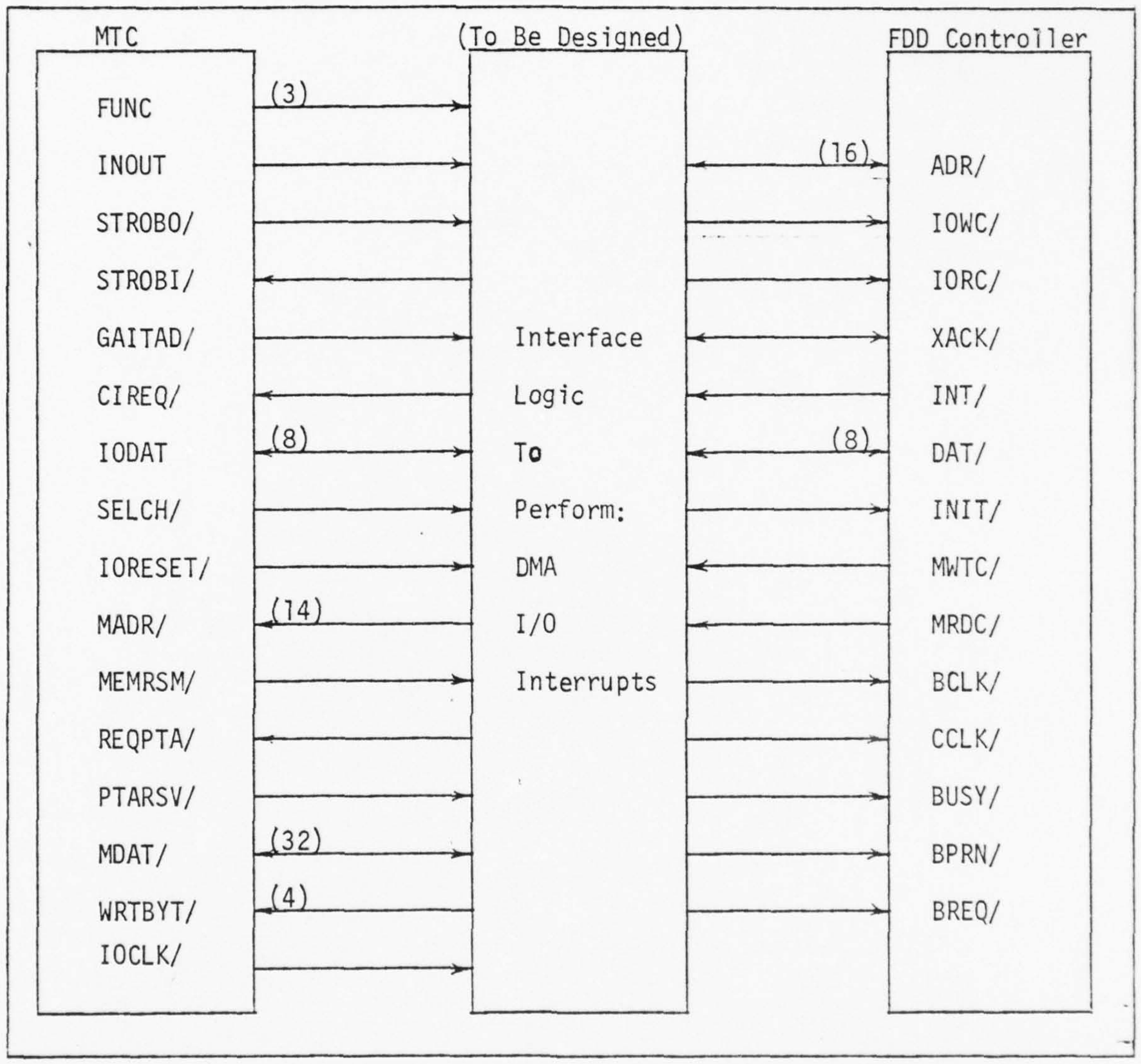
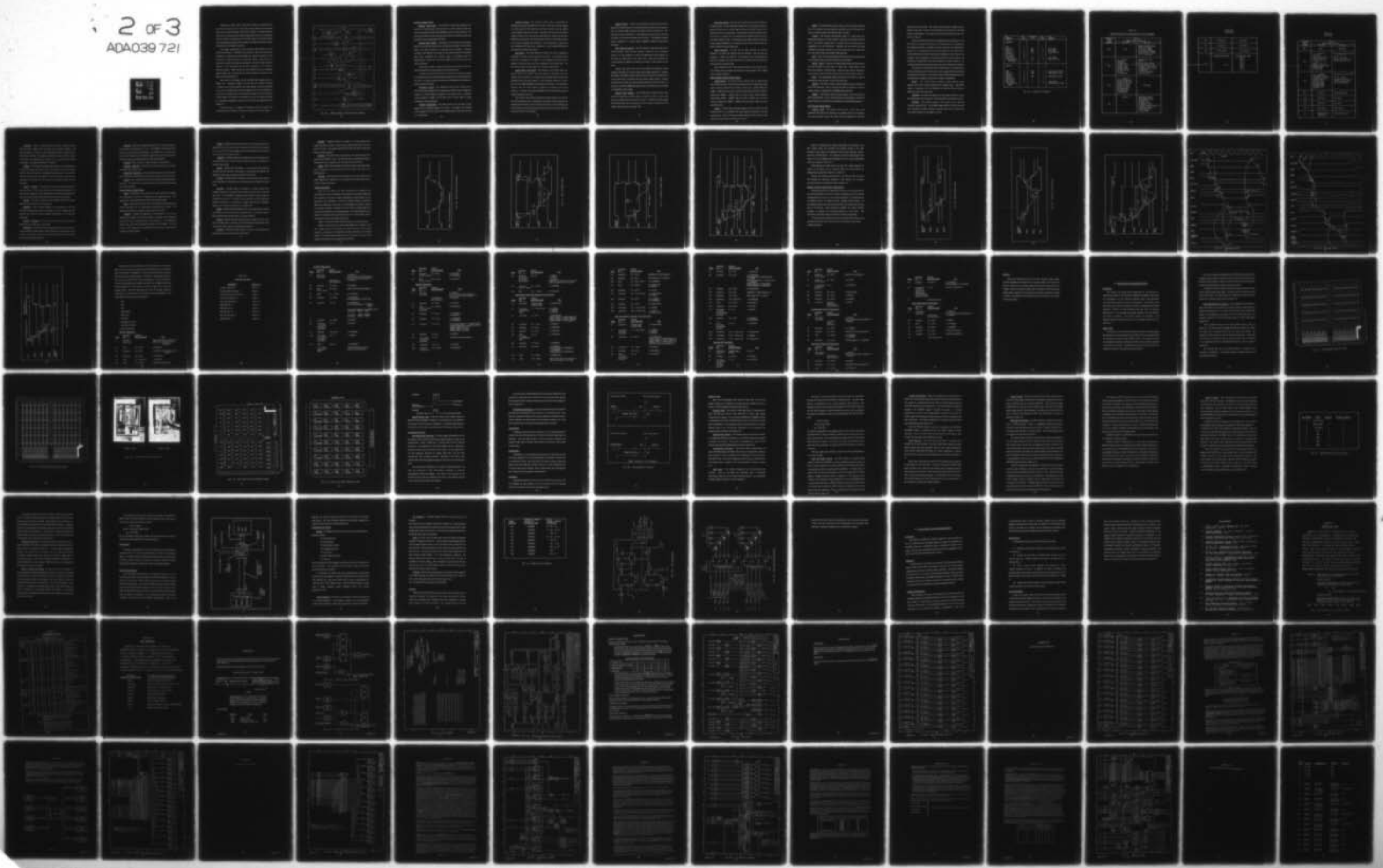
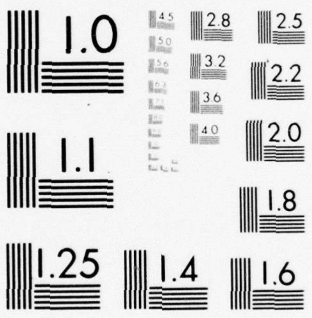


Fig. 5-2. Interface Design Bus Utilization




1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

Knowing the major actions which the interface must perform, the first step is to layout the bus structures (control, address, and data lines) which must be used to perform these actions. Figure 5-2 illustrates the bus structures to be used and the interface logic block which must be designed. The next step is to break down the interface logic into functional blocks which will explicitly generate and transform the signals inevitably required to perform all the necessary functions.

The design requirements for the interface logic follows from the needs of each system. The DOS is first studied to determine exactly what conditions are expected and what signals are required to execute an I/O instruction, a DMA operation, or an interrupt request. Next, the MTC is studied to determine what signals it generates and what it does in order to execute an I/O instruction, provide DMA capability, or to handle an interrupt request. These actions are then combined to form a composite logical design which will perform the actions required to execute an I/O, DMA, or interrupt operation.

The result of this design was the symbolic block diagram shown in Figure 5-3. Looking at Figure 5-3, one sees the overall functional breakdown, flow of data, address, and control signals. One sees the signals which will be transformed and those signals which will be generated in order for the interface logic to accomplish its function. The next sections of this chapter will discuss the function of each of the blocks and define the meaning and purpose of each of the control and data signals found in Figure 5-3.

The final step, is to design the hardware which will perform the functions laid out in Figure 5-3, which is the subject of the next chapter.

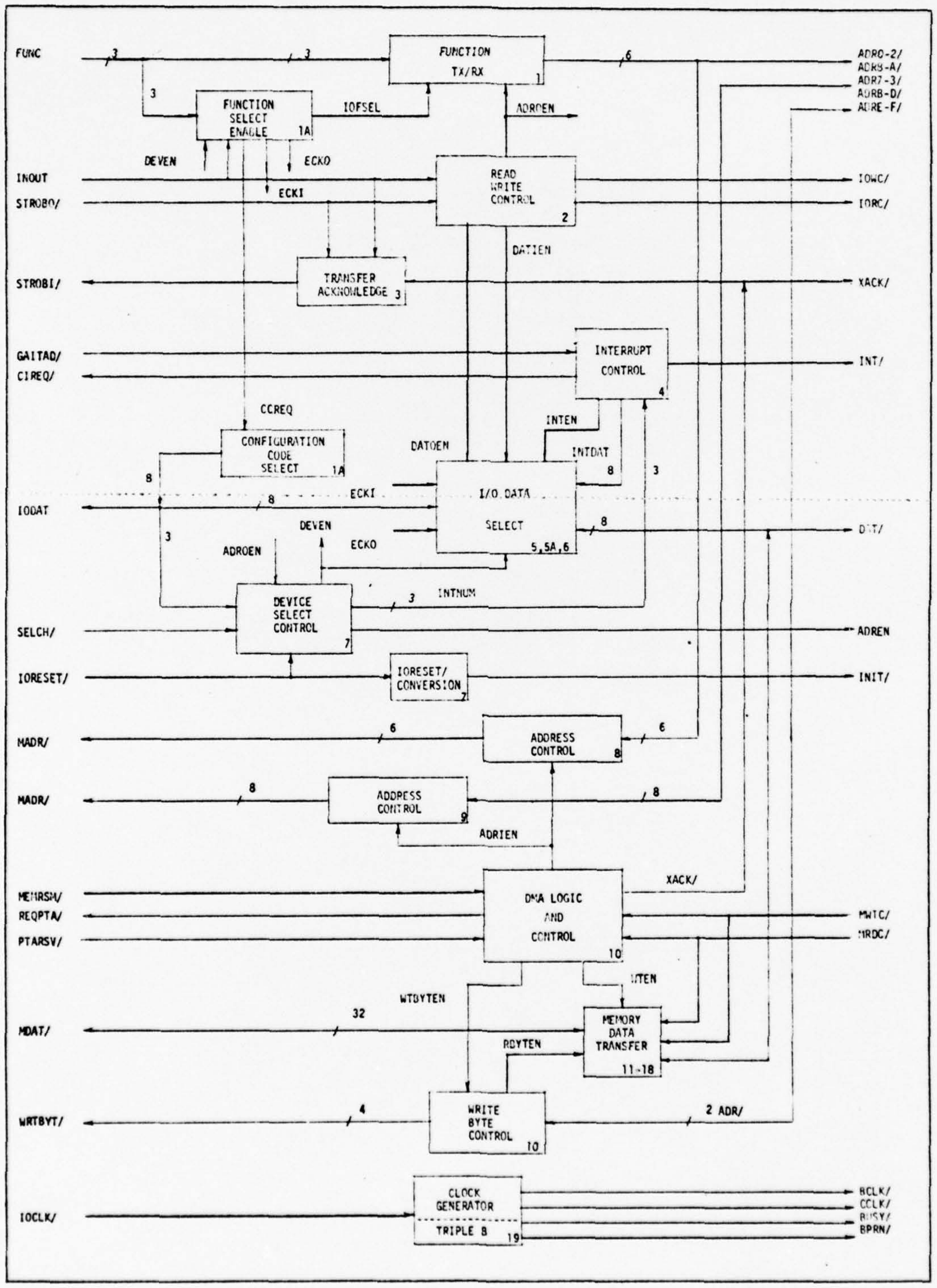


Fig. 5-3. Symbolic Design Functional Block Diagram

Interface Diagram Blocks

Function Tx/Rx Block. The Function Tx/Rx block performs the function of transmitting the function bits to the disk controller when the two control signals, IOFSEL and ADROEN, are concurrently asserted. The function information is also asserted concurrently with the device selection information and any appropriate I/O data.

Function Select Enable. The purpose of the Function Select Enable block is to decode the function bits and determine whether the interface logic or the disk controller is to perform an I/O function. The disk controller must perform seven types of I/O functions while the interface logic must accomplish two function types in accordance with the specifications. Hence, only nine of the possible 16 function codes of the MTC are utilized.

If the disk controller is required to perform an I/O function, then this block will generate a control signal to activate that action.

In addition, this block will perform one of the two functions required to be performed by the interface logic plus generate the control signals for the other. The logic necessary to perform the configuration code request is also included in this block.

Read/Write Control. The Read/Write Control block transforms the MTC's I/O control signals into the appropriate DOS I/O control signals and insures the control signals are applied at the proper time. In addition, control signals to enable the I/O data lines to either input or output information will be generated.

Transfer Acknowledge. This block converts the controller XACK/ signal into the STROBI/ signal of the MTC. The INOUT and STROBO control signals are used to insure the STROBI/ signal is asserted only after an I/O operation.

Interrupt Control. The Interrupt Control block is responsible for handling any interrupt instigated by the DOS. This block must be capable of transforming the INT/ signal from the disk controller into the CIREQ/ control signal required by the MTC. Secondly, this block must be able to supply the MTC with interrupt data information as required by the MTC GAITAD/ control signal. The interrupt data consists of the channel number of the device requesting the interrupt. The interrupt data is asserted on the multipurpose I/O data lines. Therefore, a control signal (INTEN) must be generated to enable this function.

Data Select. This block performs a major task in that it handles all operations in which the I/O data lines are used. The following major functions are enabled in this block: (1) the passing of I/O data, (2) the passing of interrupt data, and (3) the passing of echo check data. This block must not only transmit data but it must also receive data.

Device Select and Control. The function of the Device Select and Control block is to capture the device selection information and then compare that information to a known switch setting to see if its attached peripheral device has been selected. If the comparison results in a match, then a device enable (DEVEN) control signal is activated. It is imperative, however, that the control signal is applied in accordance with timing protocol. A second control signal, ADROEN, is activated to insure that the timing requirements are met.

The interrupt data required during an interrupt request is the same as the device selection data. Consequently, this block will also generate the interrupt data required by the Interrupt Control block. (For this design, the device selection and channel selection data are the same because the FDD is the only device on the channel.)

Address Control. There are two Address Control blocks but each performs the same function. The only difference between the two blocks is that one controls eight address lines while the other controls six. The function of these two blocks is to pass address information from the disk controller to the memory address bus of the MTC. The passing of address data is enabled by a control signal (ADRIEN) activated during DMA operations.

DMA Logic and Control. This block gives the disk controller direct memory access. Since the MTC provides a DMA port for a peripheral device, it is only necessary to insure the appropriate control signals are activated and deactivated at the proper time. Insuring the sequence of control signals are applied as required by the MTC is the function of this block.

Namely, this block must convert the MWTC/ and MRDC/ control signals of the DOS into a DMA request control signal, REQPTA/. Secondly, the MWTC/ and MRDC/ signals must be used to determine whether data is to be written into or read from the MTC random access memory. Finally, this block must handle any "hand-shaking" between the MTC and DOS. In particular, the MTC's MEMRSM/ signal must be transformed into the DOS's corresponding XACK/ signal.

Memory Data Transfer. The Memory Data Transfer block must provide the necessary interface logic to transform 32 MTC memory data bits into 8 DOS memory bits and vice versa. In addition, control signals (WTEN, RBYTEN) must be provided to insure the data is transferred in the proper direciton and at the proper time.

Write Byte Control. The MTC has 14 address lines while the DOS has 16 address lines. The two additional address lines of the DOS are used to determine which one of the four eight-bit bytes of the MTC memory word will be written into or read from. The function of the Write Byte Control block then is to decode the two additional DOS address lines into one of four MTC WRTBYT/ lines. The decoding of the address lines also provides control signals (RBYTEN) to the Memory Data Transfer block which are used to enable the correct 8-bit byte.

Clock Generator. In order for the disk controller to function properly, a 10 MHZ clock (BCLK) must be provided. The MTC I/O bus provides a 5 MHZ clock (IOCLK). The function of the Clock Generator block is to transform the 5 MHZ IOCLK into a 10 MHZ clock to fulfill both BCLK and CCLK requirements.

Also included with the Clock Generator block is the Triple B block. The Triple B block provides the DOS with three essential static signals (BUSY/, BREQ/, BPRN/).

Disk Controller Interface Design Signals

ADR0/-ADRF/. The disk controller address lines are bidirectional lines. Device selection and I/O function data are input to these lines while system memory address bits are output on these lines. ADR0/-ADR2/ and ADR8/ - ADRA/ lines are used to input I/O function information; ADR3/ - ADR7/ and ADRB/ - ADRF/ lines are used to input device selection information; ADR0/ - ADRD/ lines are used to output the 14 system memory address bits; ADRE/ - ADRF/ lines are used to output the write byte select data.

IOWC/. The I/O Write Command Signal is used to inform the disk controller that it is to perform the write operation specified by the I/O function bits. That is, the disk controller performs some activity on the basis of the information received from the MTC.

IORC/. The I/O Read Command is used to inform the disk controller that it is to perform the read operation specified by the I/O function bits. That is, the disk controller provides the MTC with data.

XACK/. The Transfer Acknowledge signal is a bidirectional line having two functions. The disk controller uses this line to inform the MTC that it has either accepted data or placed data on the data lines in recognition of an I/O instruction. Secondly, this line informs the disk controller that system memory has either placed data on or taken data from the memory data lines during a DMA operation.

INT/. This is the interrupt signal which the controller activates after it has completed the operation(s) specified by the IOPB(s).

DAT0/ - DAT7/. These are the eight bidirectional data lines used by the disk controller to either input or output data. With respect to the MTC, they perform a dual function. These lines are used to transmit or receive data on either the MTC I/O data bus or the MTC memory data bus.

INIT/. The initialization signal, activated by the MTC, is used to initialize the disk controller to known internal states.

MWTC/. The Memory Write Command signal is used to activate a DMA write operation. That is, the disk controller will transfer n words to system memory as specified by the IOPB(s) being executed.

MRDC/. The Memory Read Command signal is used to activate a DMA read operation. That is, the disk controller will transfer n words from system memory to the diskette as specified by the IOPBs being executed

MTC Interface Design Signals

Function Lines. The function lines provide a 3-bit code which determine what type of I/O operation the peripheral device is to perform. The three function lines of the MTC must be mapped into the three function bits of the DOS. The reader should reference Chapter III, for details on the MTC function bits and Chapter IV, for details on the DOS function requirements. The details of the uses of the function bits are also shown in Table VII.

Since only the Echo Check and Configuration Code functions of the MTC need to be preserved, as noted in the specifications, the mapping of the functions become a straight one-to-one conversion as illustrated in Figure 5-4. The complexity of the interface design is to the extent that it must include the Echo Check function and Configuration Code. That is, the interface must be capable of performing the Echo Check function and returning a Configuration Code when requested by system software. All remaining functions will be interpreted by the driver software for the DOS.

The purpose of the Function Select Enable block is to determine the function code and provide the necessary control signals which will activate the appropriate hardware. The Function Select Enable block generates the following control signals: ECKI, ECKO, CCREQ1, and IOFSEL.

INOUT. This control signal specifies an input operation when logically low and an output operation when logically high. The INOUT signal in conjunction with the STROBO/ and Function lines activates a particular type of I/O instruction.

STROBO/. The Strobe Out Signal is the last signal activated to insure the execution of an I/O instruction. (See also INOUT and Function Lines.)

STROBI/. The Strobe In signal is the counter part to the disk controller signal XACK/. The STROBI/ signal informs the MTC that the disk controller has either accepted or placed data on the I/O data lines. The timely reception of this signal is critical.

MTC COMMAND	MTC CODE	RESPONSE BY	FDC CODE	FDC COMMAND
(OUTPUT)				
Data	0	-	-	-
Control 1	1	FDC	1	Write MAL
Control 2	2	FDC	2	Write MAU
Control 3	3	FDC	3	Stop Disk Op
Position 1	4	-	-	-
Position 2	5	-	-	-
Echo Check	6	I	6	Echo Check
Interprocessor Interrupt	7	FDC	7	Reset Disk Op
(INPUT)				
Data	0	FDC	0	Read Subsys Status
Status 1	1	FDC	1	Read Result Type
Status 2	2	-	-	-
Status 3	3	FDC	3	Read Result Byte
Position 1	4	-	-	-
Position 2	5	-	-	-
Echo Check	6	I	6	Echo Check
Configuration Code	7	I	7	Configuration Code

Fig. 5-4. Function Bit Mapping

Table VII

Function Coding And Associated Data Line Assignments

FUNCTION CODE (BINARY)	INPUT INOUT LINE = LOGICAL 0	
	MTC USE	DOS USE
000	Data	Read Subsystem Status 2^0-Drive 0 Ready 2^1-Drive 1 Ready 2^2-Interrupt Pending 2^3-Controller Present 2^4-2^7: Not Used
001	Status 1 2^0-Ready 2^1-Not Busy 2^2-Man. Input 2^3-End of Op 2^4-Data Ready 2^5-Data Request 2^6-Error 2^7-Interrupt	Read Result Type 2^0-2^1: 01-I/O Cplt w/error 00-I/O Cplt w/error 10-Result byte contains diskette ready status 11-Reserved 2^2-2^7-Block number used only w/01
010	Status 2 2^0-Write Enabled 2^1-Lost Data Error 2^2-End of Record 2^3-End of File 2^4-End of Media 2^5-Unassigned 2^6-Unassigned 2^7-Reserved	Not Used
011	Status 3	Read Result Byte 2^0-Deleted Record 2^1-CRC Error 2^2-Seek Error 2^3-Address Error 2^4-Data Overrun/Underrun 2^5-Write Protect 2^6-Write Error 2^7-Not Ready

Table VII
(continued)

100	Position 1	Not Used
101	Position 2	Not Used
110	Echo Check	Echo Check
111	Configuration Code	Configuration Code 2^0 - 1 2^1 - 0 2^2 - 1 2^3 - 1 2^4 - 0 2^5 - 1 2^6 - 0 2^7 - 0

Table VII
(continued)

FUNCTION CODE (BINARY)	OUTPUT INOUT LINE = LOGICAL 1	
	MTC USE	DOS USE
000	Data	Not Used
001	Control 1 2^0-Clear 2^1-Sel Binary Data 2^2-Sel Data Int 2^3-Sel End of Op Int 2^4-Sel Error Int 2^5-Desel Manual Int 2^6-Not Used 2^7-Not Used	Write Memory Addr Lower 2^0-2^7: 8 least significant digits of memory address word
010	Control 2 2^0-Reverse Motion 2^1-Execute Function 2^2-Skip Record 2^3-Skip File 2^4-Unassigned 2^5-Unassigned 2^6-Reset Media 2^7-Write	Write Memory Addr Upper 2^0-2^7: 8 most significant digits of memory address word and start command
011	Control 3	Stop Diskette Op
100	Position 1	Not Used
101	Position 2	Not Used
110	Echo Check	Echo Check
111	Interprocessor Interrupt	Reset Diskette Op

GAITAD/. After an interrupt request has been recognized by the MTC, the MTC responds with this signal, Gate Interrupt Address. The GAITAD/ signal is a request for the interrupting device to respond with its device select address. This address information allows the MTC to know which device is requesting the interrupt so it can service the request accordingly. This signal is subsequent to CIREQ/.

CIREQ/. The channel Interrupt Request signal is the counterpart to the disk controller's INT/ signal. When this signal is activated, it informs the MTC that some device is requesting interrupt service. At this point, the MTC does not know which device. In order to find out which device is requesting the interrupt, the MTC asserts the GAITAD/ active low (see GAITAD/).

IODAT0 - IODAT7. The eight I/O data lines are bidirectional and multi-functional. The I/O data lines serve the following functions: (1) receives the configuration code assignments, (2) receives and transmits I/O data, (3) transmits device select data and (4) receives interrupt data.

SELCH/. The Select Channel control signal activates the device selection data coming off the I/O data lines.

IORESET/. The I/O Reset signal is the counterpart of the disk controller's Initialization signal. This signal is used to initialize the disk controller and reset the device selection information in the interface design logic.

MADR0/ - MADRD/. The 14 MTC memory address lines are used to address any of the 16K words in the RAM.

MEMRSM/. The Memory Resume signal provides the second function of the controller's XACK/ signal. When activated, this signal informs the controller that system memory has accepted or placed data on the memory data bus during a DMA operation.

REQPTA/. Whenever a peripheral device desires to perform DMA, it must activate the Request Port A line. The REQPTA/ signal informs the ACU that a device has requested DMA. If the DMA port is not busy, then the ACU will respond with a PTARSV/ signal which will then allow direct memory access by the peripheral device. (See also PTARSV/).

PTARSV/. This is the Port A Reserved signal. (See REQPTA/.).

MDAT00/ - MDATIF/. The 32 memory data lines are used to transmit and receive 8-bit bytes of data.

WRTBYT0/- WRTBYT3/. The four Write Byte lines are used to select one of the four 8-bit bytes of data which will be written into or read from.

IOCLK/. The 5 MHZ I/O clock is used to synchronize appropriate I/O operations. The IOCLK will be used to generate the 10 MHZ clocks needed by the disk controller.

Internal Interface Design Signals

ECKO. ECKO (Echo Check Out) is the control signal which enables the transmission of one bit of information over eight data lines. The information is transmitted from the MTC to the DOS controller.

ECKI. ECKI (Echo Check In) is the control signal which enables the return transmission of the information transmitted with ECKO. The information passes from the DOS controller back to the MTC.

CCREQ/. CCREQ/ (Configuration Code Request/) is the control signal which requests the interface design to respond with information telling what type of peripheral device is attached to it. A configuration code of $2D_{16}$ has been arbitrarily selected for this design. The only criteria to be considered when selecting this code is that no other device can have the same number.

IOFSEL. IOFSEL (I/O Function Select) is the control signal which is generated for every function code the floppy disk controller must respond to. It is one of two control signals which must be present to activate the function lines onto the address lines.

ADROEN. ADROEN (Address Out Enable) is the control signal which allows information on the MTC function and IO data lines to be asserted on the DOS address lines.

DEVEN. DEVEN (Device Enable) is a control signal which indicates the DOS has been selected. This signal, in conjunction with IOFSEL and ADROEN, is also used to activate the MTC function lines.

DATIEN. DATIEN (Data In Enable) is a control signal which enables Echo Check data to be passed from the DOS data lines to the MTC I/O data lines.

DATOEN. DATOEN (Data Out Enable) is a control signal which enables information to be passed from the MTC I/O data lines to the DOS data lines. The DATOEN is generated when INOUT is active high and STROBO/ is active low. The INOUT control signal of the MTC determines whether information is to be input or output while the MTC STROBO/ signal is used to activate the lines at the proper time.

INTEN. INTEN (Interrupt Enable) is the control signal which enables the INTDAT lines to be asserted on the IODAT lines. The INTEN signal is generated whenever the DOS wants to interrupt the MTC.

INTDAT. INTDAT (Interrupt Data) consists of eight data lines. Only one of the eight lines is activated. The MTC decodes all eight IO data lines to determine which channel is requesting the interrupt.

INTNUM. INTNUM (Interrupt Number) consists of three lines which provide device and channel selection information.

ADRIEN. ADRIEN (Address in Enable) is a control signal which enables the DOS controller to pass memory address information from the DOS to the MTC. This signal is generated whenever the DOS is required to perform a DMA operation.

WBYTEN. WBYTEN (Write Byte Enable) is the control signal which activates the WRTBYT lines. The WBYTEN signal is generated whenever information is to be transferred from the DOS to the MTC RAM.

WTEN. WTEN (Write Enable) is the control signal which allows eight-bits of information to be passed from the MTC memory data lines to the DOS data lines.

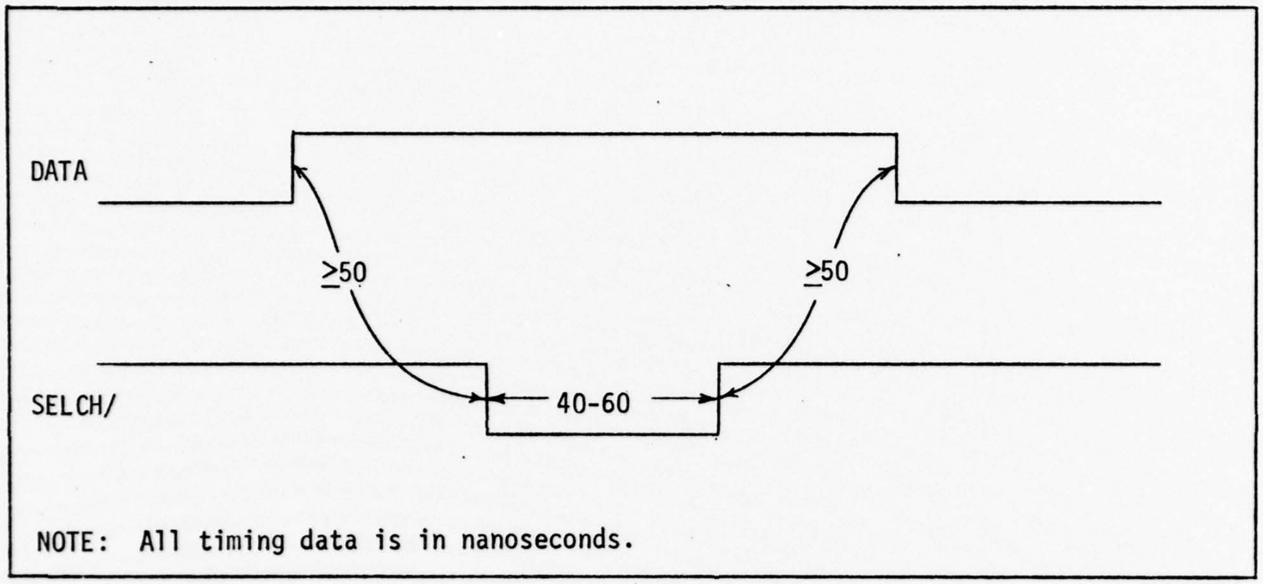
RBYTEN. RBYTEN (Read Byte Enable) is the control signal which allows eight-bits of information to be passed from the DOS data lines to the MTC memory data lines.

Timing Relationships

Now that the system has been partitioned and organized into functional blocks and the necessary and essential control signals identified, the mandatory and critical timing relationships of these signals must be determined and developed to meet the interface design requirements (DMA, I/O, Interrupt). This action is accomplished by carefully studying the material presented in Chapters III and IV and the timing diagrams of each system presented in the following pages. The timing diagrams which represent composite timing relationships are the critical design requirements which the interface design must adhere.

Figure 5-5 shows the select channel and device timing relationship.

Figure 5-6 illustrates the timing relationships of the signals in the MTC system which are activated and required during an I/O output operation, while Figure 5-7 illustrates the same information for the DOS system. From these timing relationships, the composite I/O output timing diagram (see Figure 5-8) was designed and developed.



NOTE: All timing data is in nanoseconds.

Fig. 5-5. Device Select Timing Diagram

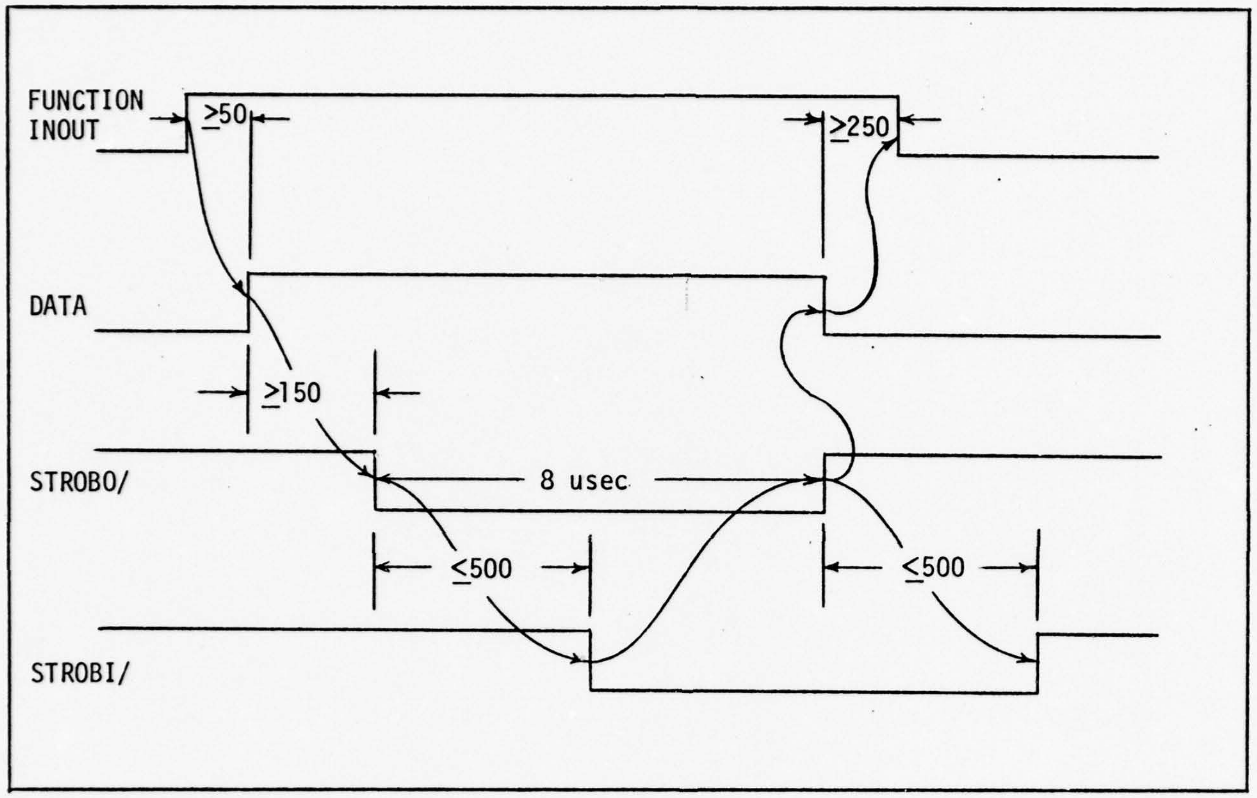


Fig. 5-6. MTC I/O Output Timing

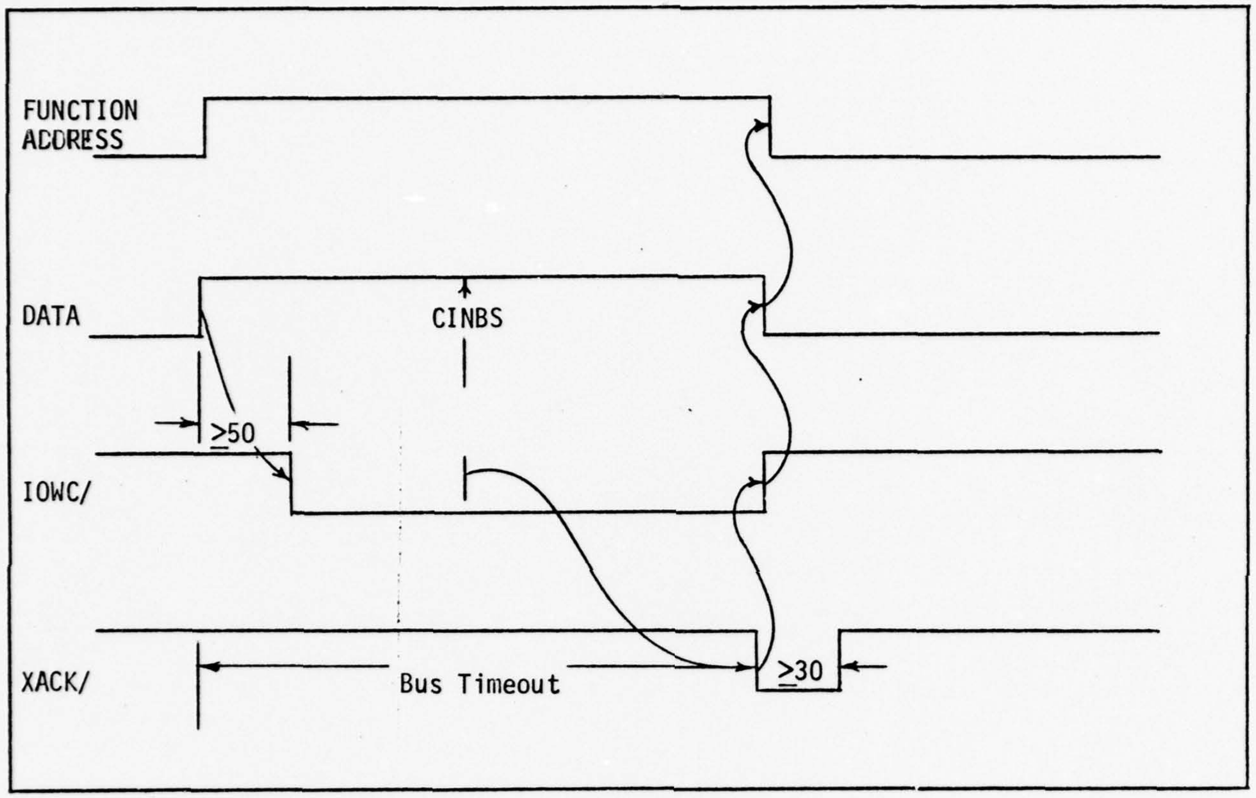


Fig. 5-7. Controller Output Timing

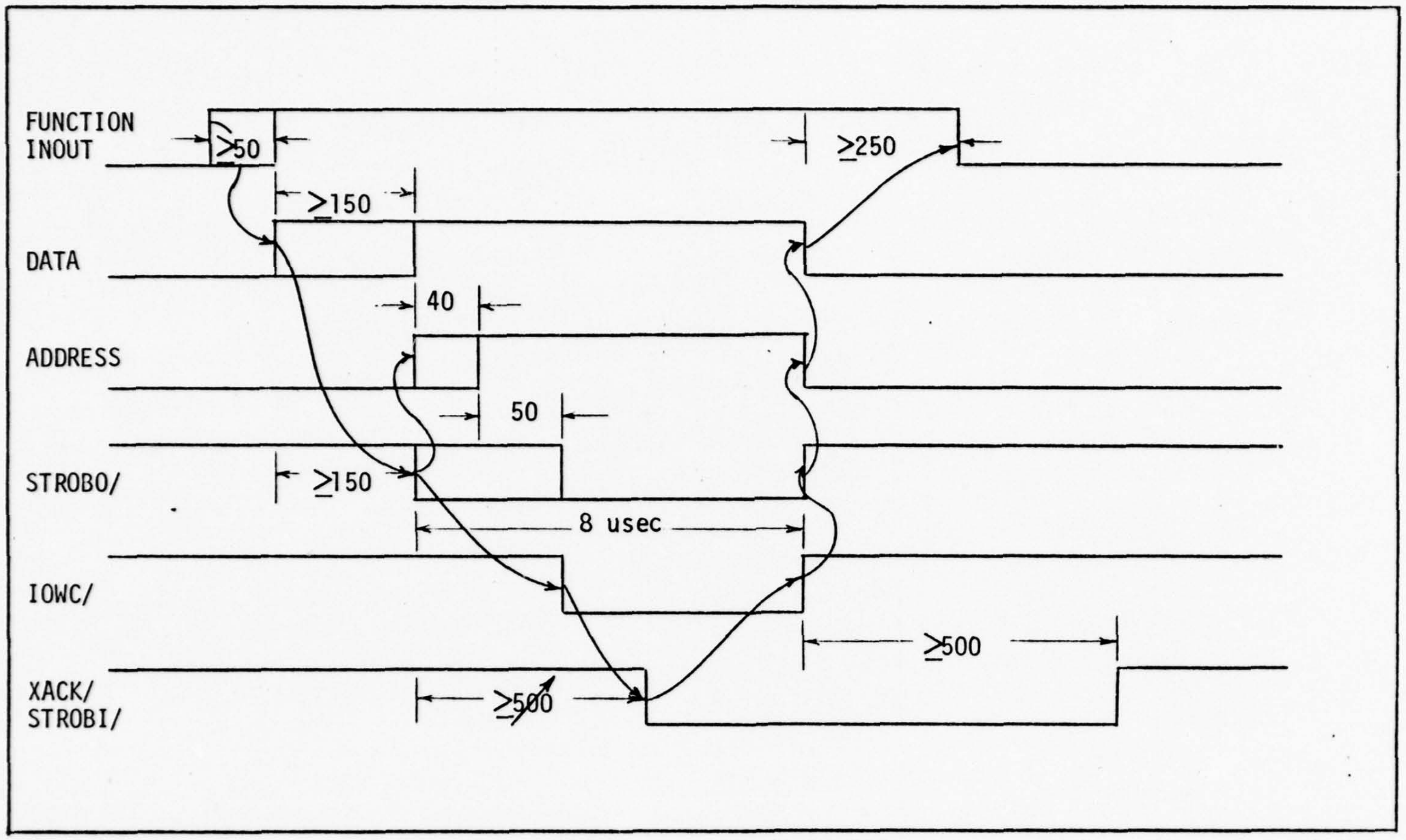


Fig. 5-8. Composite I/O Output Timing

Figure 5-9 illustrates the timing relationships of the signals in the MTC system which are activated and required during an I/O input operation, while Figure 5-10 illustrates the I/O input operation requirements for the DOS system. The composite I/O input timing diagram (see Figure 5-11) was designed and developed from the timing relationships presented in Figures 5-9 and 5-10.

Understanding the DOS system and the MTC timing diagrams of Figure 3-8 and Figure 3-9, the composite DMA I/O timing diagram was designed and is presented in Figures 5-12 and 5-13.

Finally, from information presented in the DOS and MTC literature and studying the schematic diagrams of the systems, the composite interrupt timing was developed as shown in Figure 5-14.

Register Transfer Language (RLT) Specification

Having partitioned and organized the system into functional blocks, and having developed the critical timing relationships, the RTL description/specification of each unit may now be developed. The development of the design, in terms of register transfer language (RTL), defines the information transfer and control operations that must be performed at each step of the process. These statements establish the control sequence necessary to regulate the functions allocated to each block. This information will then be used to develop the hardware logic design.

In the final system realization section, the operational characteristics specified by the RTL will be used to produce the hardware logic design timing information.

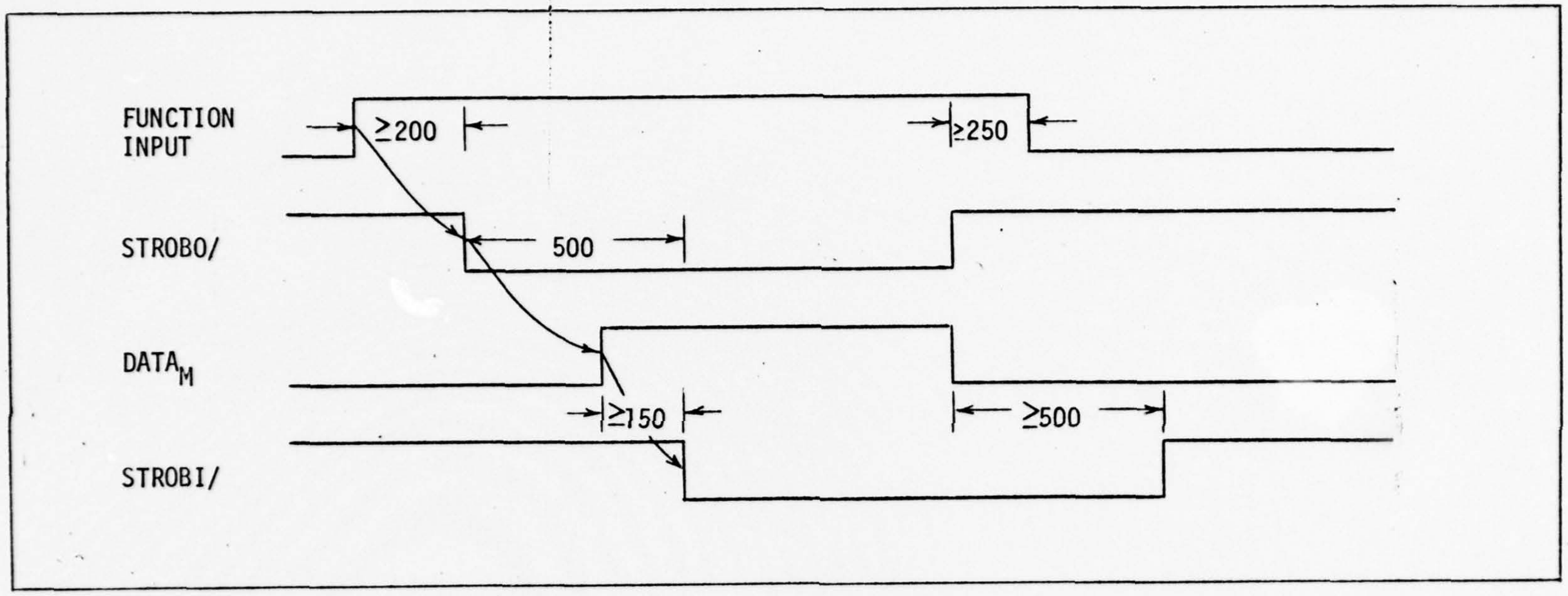


Fig. 5-9. MTC I/O Input Timing

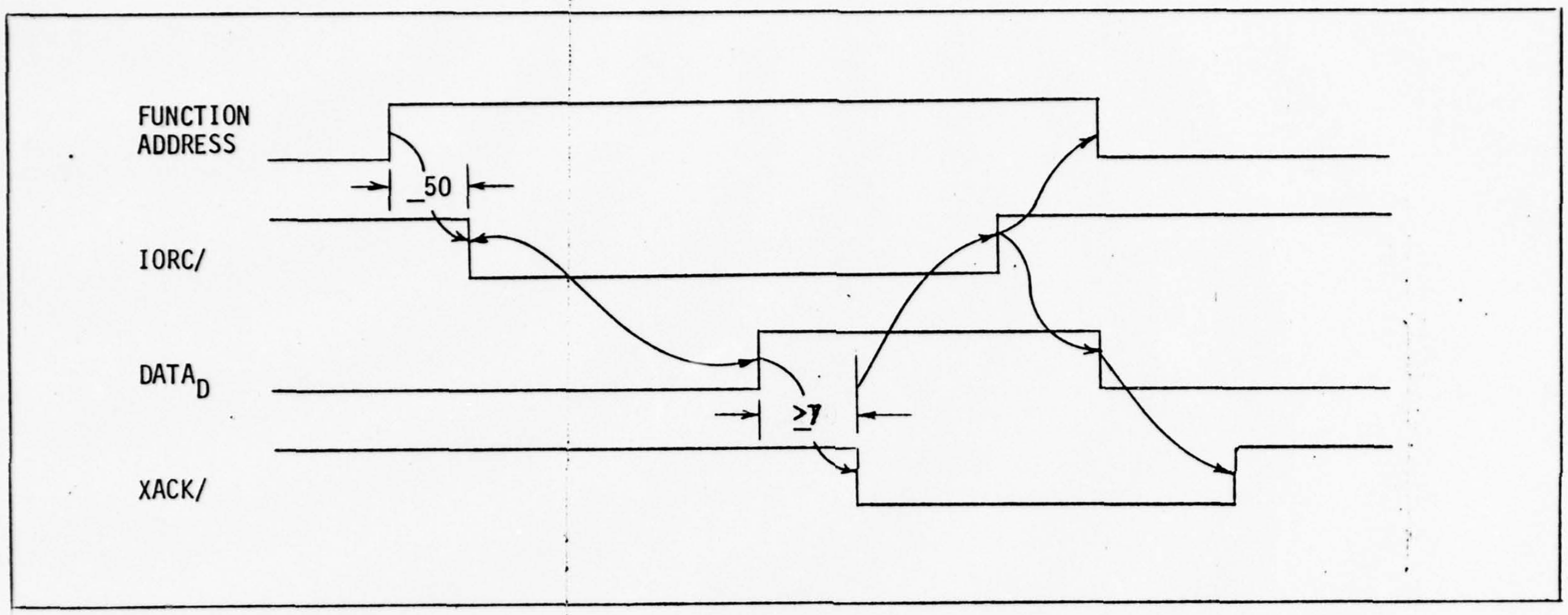


Fig. 5-10. Controller Input Timing

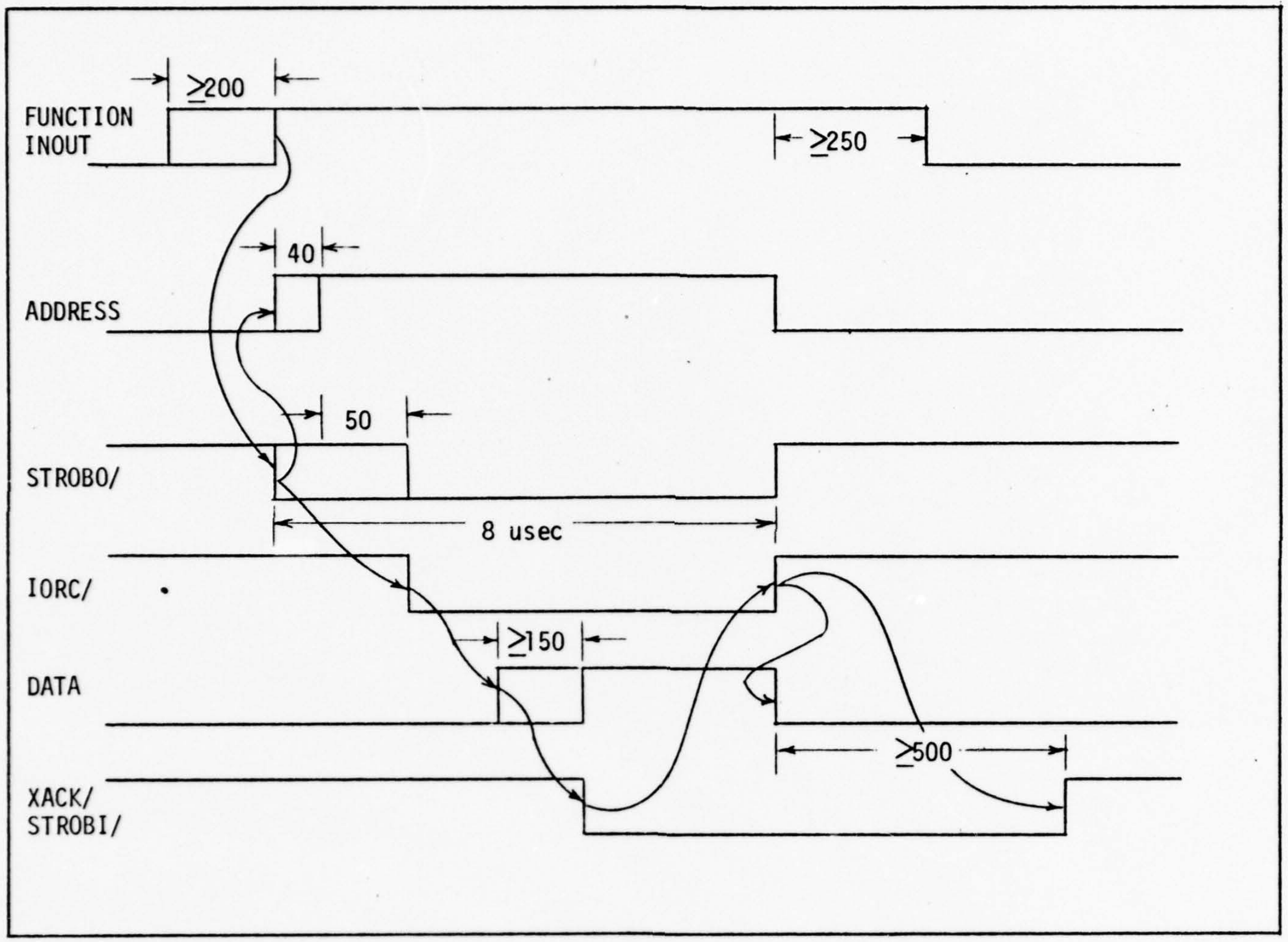


Fig. 5-11. Composite I/O Input Timing

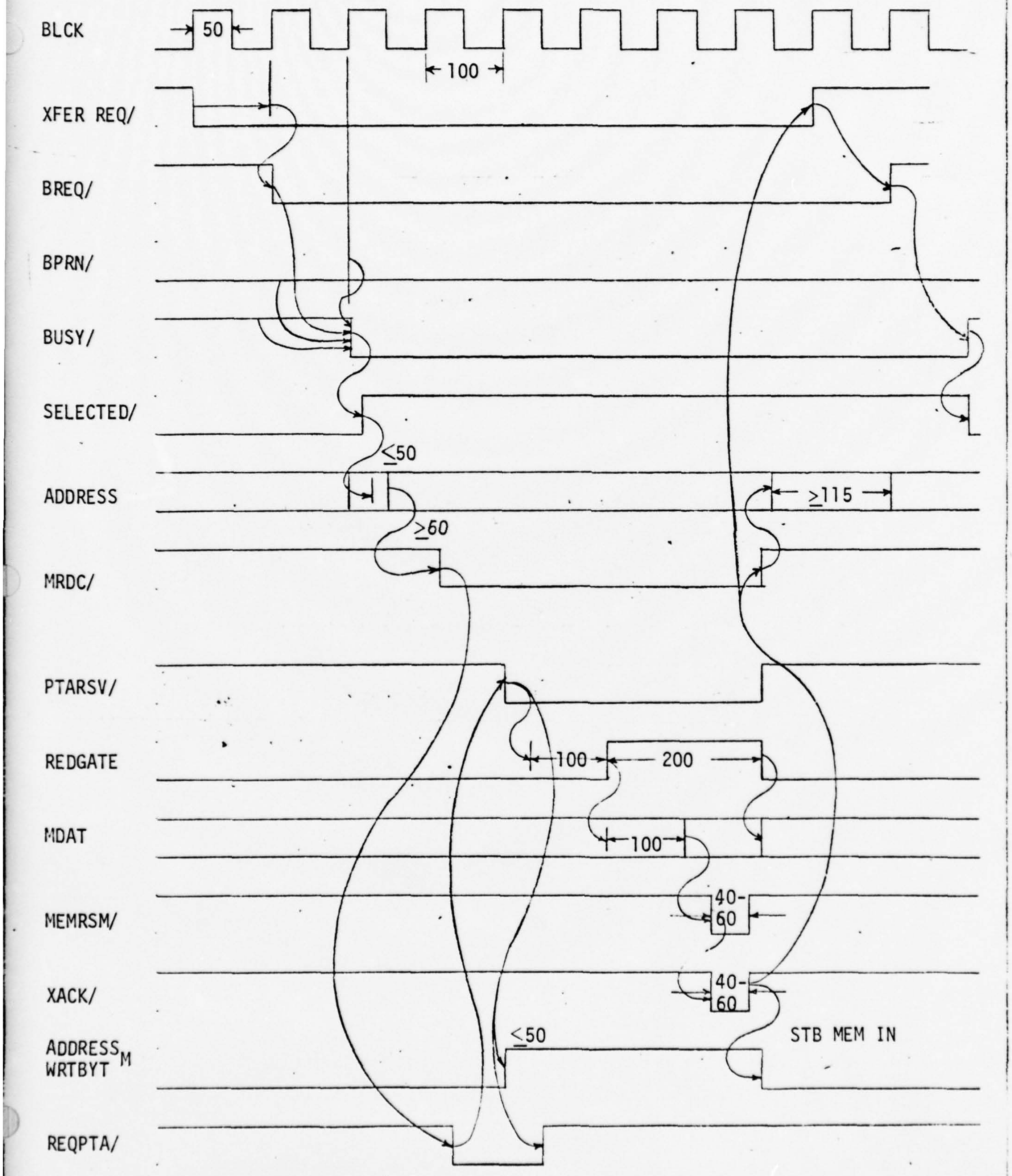


Fig. 5-12. DMA Output Timing

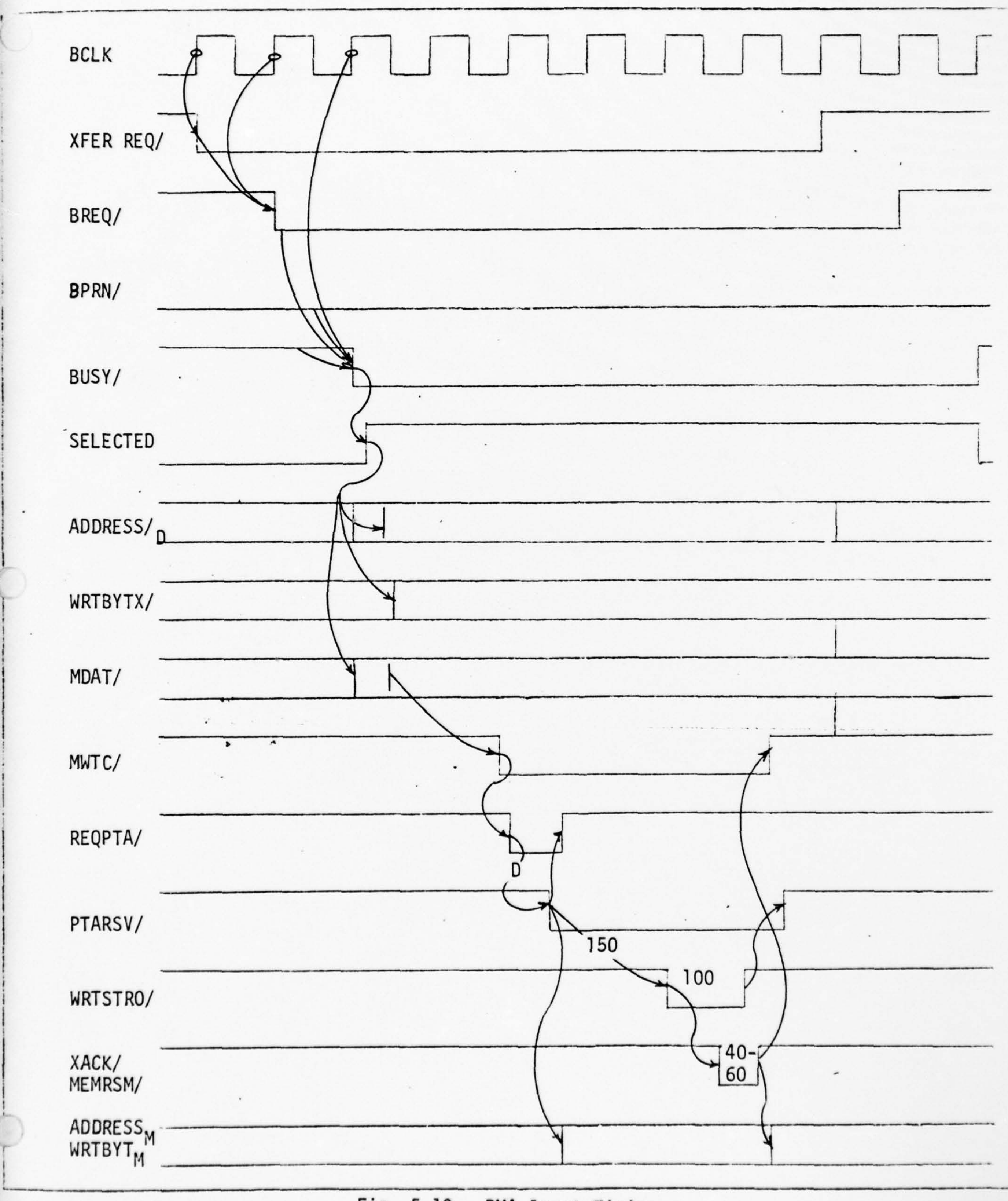


Fig. 5-13. DMA Input Timing

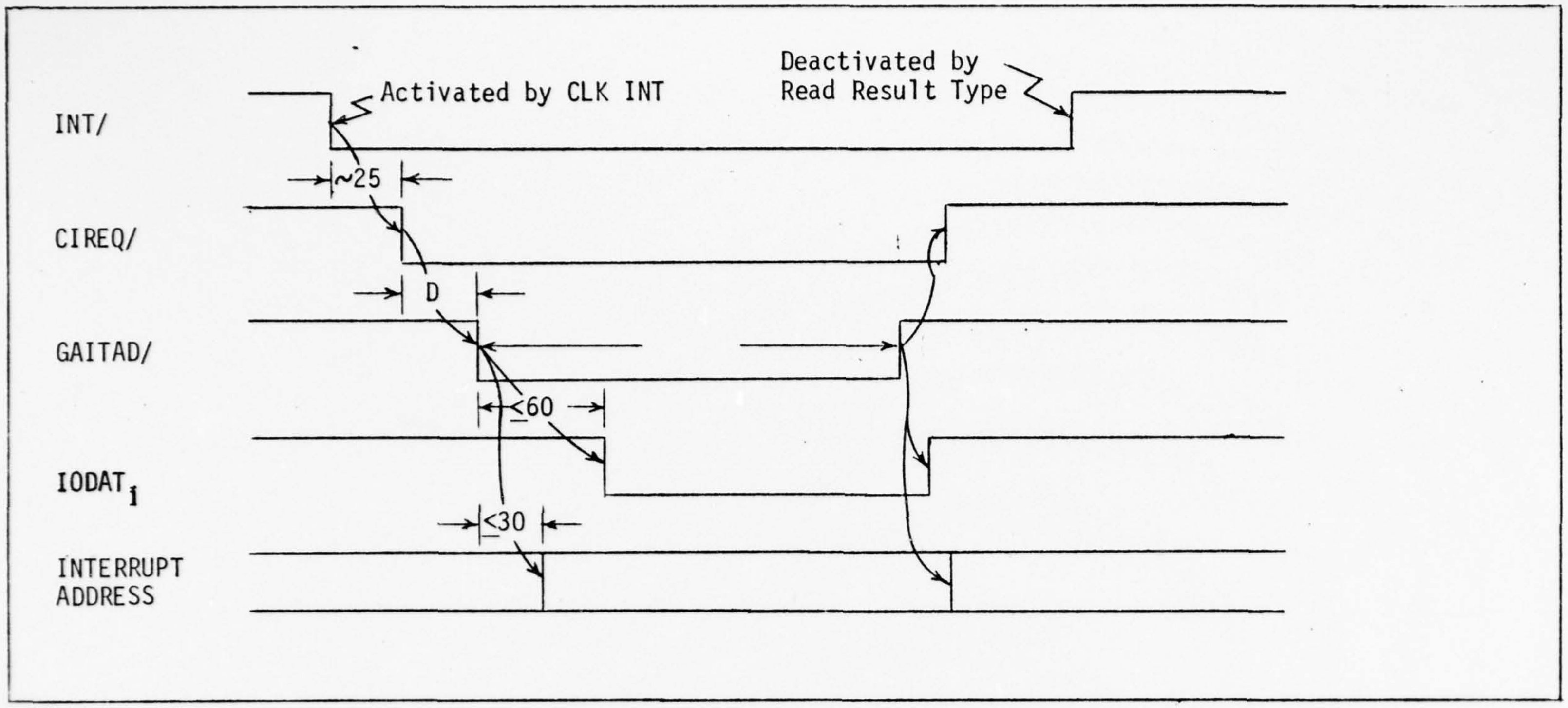


Fig. 5-14. Interrupt Timing

Included with the RTL description is timing information. Gate delay, which may or may not be critical, delays the execution of an operation. Since the interface is designed for worst case timing conditions and gate delay may vary from approximately 3 to 25 nsec, a typical gate delay of 10 nsec is used to determine the timing relationships in the RTL description. In the RTL description, any number associated with a register name (e.g., IO Data Register 5) is the number of a block in Figure 5-3 in which the register exists. The RTL descriptions for each of the logic operations listed below are documented in the subsequent ollowing paragraphs: (See Table VIII for the list of interface registers.)

- SEL
- OUT
- INP
- DMA Output
- DMA Input
- Interrupt
- Configuration Code
- Echo Check Output
- Echo Check Input

SEL RTL Description

TIME	DEFINED BY	TIMING RELATIONSHIP	RTL
T1	MTC SEL Instruction		IODAT (8) $\rightarrow$ IO Data Register 5 IODAT (3) $\rightarrow$ Device Select Register 7
T2	MTC	T1 + 60 ns	0 $\rightarrow$ SELCH/
T3	Interface	T2 + 10 ns	INTNUM (3) $\rightarrow$ Interrupt Control Register 4
T4	Interface	T3 + 30 ns	1 $\rightarrow$ DEVEN *
T5	MTC	T2 + (40-60 ns)	1 $\rightarrow$ SELCH/
T6	MTC	T5 + 50 ns	IODAT (8) Deactivated

Table VIII

Interface Registers

DESIGNATOR	Figure 5-2
IO Data Register 5	Block 5
Device Select Register 7	Block 7
Interrupt Control Register 4	Block 4
Func Tx/Rx Register 1	Block 1
Func Sel Register 1A	Block 1A
ADDR Register 8	Block 8
ADDR Register 9	Block 9
DMA Register 10	Block 10
Mem Data Register 11	Block 11
DMA Register 11	Block 11

OUT RTL Description

TIME	DEFINED BY	TIMING RELATIONSHIP	RTL
T1	MTC OUT Instruction		1 → INOUT Func (3) → Func Tx/RX Register 1 Func (3) → Func Sel Register 1A
		Activated by SEL instruction	1 → DEVEN
T2	Interface	T1 + 30 ns	1 → IOFSEL
T3	MTC (≥T1 + 50 ns)	T2 + 20 ns	IODAT (8) → IO Data Register 5
T4	MTC	T3 + 150 ns	0 → STROBO/
T5	Interface	T4 + 20 ns	1 → DATOEN IO Data Register 5 → DAT/ (8)
T6	Interface	T5 + 20	1 → ADROEN Func Tx/Rx Register 1 → ADRO/-ADR2/ Func Tx/Rx Register 1 → ADRB/-ADRA/ Activated ADR3/ - ADR7/ Activated ADRB/ - ADRD/ Activated ADRE/ - ADRF/
T7	Interface	T6 + 50 ns	0 → IOWC/
T8	Controller D = a delay function of controller	T7 + D	0 → XACK
T9	Interface	TB + 60 ns	0 → STROBI/
T10	Controller D = a delay function of controller	T7 + D	1 → IOWC/
T11	MTC	T10 + D	1 → STROBO/
	D = a delay function of MTC		Address Lines Deactivated Data Lines Deactivated

TIME	DEFINED BY	TIMING RELATIONSHIP	RTL
T12	Interface (≤ T10 + 500 ns)	T11 + 20 ns	1 → STROBI/ 0 → ADROEN
T13	MTC (≤T10 + 250 ns)	T10 + 250 ns	0 → FUNC (3)

INP RTL Description

TIME	DEFINED BY	TIMING RELATIONSHIP	RTL
T1	MTC INP Instruction		0 → INOUT Func (3) → Func Tx/Rx Register 1 Func (3) → Func Select Register 1A
		Activated by SEL instruction	1 → DEVEN
T2	Interface	T1 + 30 ns	1 → IOFSEL
T3	MTC (8 microsecond time-out)	T1 + 200 ns	0 → STROBO/
T4	Interface	T3 + 10 ns	0 → STROBO/ 1 → INOUT/
T5	Interface	T4 + 10 ns	1 → ADROEN 1 → STROBO
T6	Interface	T5 + 40 ns	Func Tx/Rx Register 1 → ADR0/-ADR2/ Func Tx/Rx Register 1 → ADRB/-ADRA/ ADRB/-ADRD/ activated ADR3/-ADR7/ activated ADRE/-ADRF/ activated
T7	Interface	T6 + 50	0 → IORC/
T8	D = a delay function of controller	T7 + D	DAT/(8) → IO Data Register 5
T9	Controller	T8 + 7 ns	0 → XACK/
T10	Interface (≥ T8 + 150)	T9 + 150 ns	0 → STROBI/

TIME	DEFINED BY	TIMING RELATIONSHIP	RTL
T11	Interface	T10 + D	$1 \rightarrow$ IORC/
	D = a delay		$1 \rightarrow$ DAT/
	determined		$1 \rightarrow$ XACK/
	by controller		STROBO/ Deactivated = 8 micro sec
			ADR/ Deactivated = 8 micro sec
T12	Interface	T11 + 120 ns	$1 \rightarrow$ STROBI/
	($\leq$ T11 + 500 ns)		
T13	MTC	T11 + 250 ns	$0 \rightarrow$ Func (3)

DMA Output (Read From Memory) RTL Description

TIME	DEFINED BY	TIMING RELATIONSHIP	RTL
T1	Controller	Negative Edge	$0 \rightarrow$ XFER REQ/
		of XFER REQ/	
T2	Controller	T1 + 1 Clock Cycle	$0 \rightarrow$ BREQ/
	Activated By		$0 \rightarrow$ BPRN/
	BCLK		$0 \rightarrow$ BUSY/
T3	Controller	T2 + 30 ns	$1 \rightarrow$ SELECTED
			ADRO/-ADRS/ $\rightarrow$ ADDR Register 8
			ADRG/-ADRO/ $\rightarrow$ ADDR Register 9
			ADRE/-ADRF/ $\rightarrow$ DMA Register 10
T4	Controller	T3 + 60 ns	$0 \rightarrow$ MRDC/
T5	Interface	T4 + 40 ns	$1 \rightarrow$ RBYTEN
T6	Interface	T5 + 30 ns	$0 \rightarrow$ REQPTA/
T7	MTC	T6 + D	$0 \rightarrow$ PTARSV/
	D = Delay		
	determined by		
	ACU		
T8	Interface	T7 + 30 ns	$1 \rightarrow$ REQPTA
T9	Interface	T8 + 20 ns	$1 \rightarrow$ ADRIEN1
			ADDR Register 8 $\rightarrow$ MADR/(6)
			$1 \rightarrow$ ADRIEN2
			ADDR Register 9 $\rightarrow$ MADR/(8)
T10	MTC	T7 + 100 ns	$1 \rightarrow$ REDGATE
T11	MTC	T10 + 100 ns	MDAT (32) $\rightarrow$ Mem Data Register 11
			Mem Data Register 11 $\rightarrow$ DAT/

TIME	DEFINED BY	TIMING RELATIONSHIP	RTL
T12	Interface	T11 + 50 ns	MEMRSM/ → DMA Register 11
T13	Interface	T12 + 20 ns	DMA Register 11 → XACK/
T114	MTC	T112 + (40 - 60 ns)	1 → MEMRSM/
T15	MTC	T10 + 200 ns	0 → REDGATE 0 → MDAT
T16	Interface	T14 + 20 ns	1 → XACK
T17	Controller	T14 + 1 clock cycle	1 → XFER REQ/ 1 → MRDC/
T18	Interface	T16 + 20 ns	0 → ADRIEN1 0 → ADRIEN2
T19	Interface	T17 + 30 ns	0 → RBYTEN(4)
T20	Controller	T17 + 1 clock cycle	1 → BREQ/
T21	Controller	T20 + 1 clock cycle	1 → BUSY/ 0 → SELECTED

DMA Input (Write to Memory) RTL Description

TIME	DEFINED BY	TIMING RELATIONSHIP	RTL
T1	Controller	Negative edge of XFER REQ/	0 → XFER REQ/
T2	Controller Activated By BCLK/	T1 + 1 clock cycle	0 → BREQ/ 0 → BPRN/ 0 → BUSY/
T3	Controller	T2 + 30 ns	1 → SELECTED ADR0/-ADR51 → ADDR Register 8 ADR6/-ADRD/ → ADDR Register 9 ADRE/-ADRF/ → DMA Register 10 DAT/(8) → Mem Data Register 11
T4	Controller	T3 + 50 ns	0 → MWTC/
T5	Interface	T4 + 70 ns	0 → REQPTA/
T6	MTC D = Delay determined by ACU	T5 + D	0 → PTARSV/

TIME	DEFINED BY	TIMING RELATIONSHIP	RTL
T7	Interface	T6 + 30 ns	1 → REQPTA/
T8	Interface	T7 + 20 ns	1 → WTBYTEN DMA Register 10 → WRTBYT/(4) 1 → WTEN Mem Data Register 11 → MDAT (32) 1 → ADRIEN ADDR Register 8 → MADR/(6) ADDR Register 9 → MADR/(8)
T9	Interface	T6 + 150 ns	0 → WRTSTRO/
T10	Interface	T9 + 50 ns	MEMRSM/ → DMA Register 10
T11	Interface	T10 + 20 ns	DMA Register 10 → XACK/
T12	Interface	T10 + (40-60 ns)	1 → MEMRSM/
T13	MTC	T9 + 100 ns	1→WRTSTRO/
T14	Interface	T12 + 20 ns	1 → XACK/
T15	Controller D = delay determined by controller	T11 + D	1 → MWTC/
T16	Interface	T12 + 30 ns	0 → ADRIEN 0 → WTBYTEN
T17	MTC D = delay determined by ACU	T13 + D	1 → PTARSV/
T18	Controller	T15 + 1 clock cycle	1 → XFER REQ/
T19	Controller	T18 + 1 clock cycle	1 → BREQ/
T20	Controller	T19 + 1 clock cycle	0 → SELECTED

Interrupt RTL Description

TIME	DEFINED BY	TIMING RELATIONSHIP	RTL
T1	DOS Micro-program	Negative edge of INT/	0 → INT/
T2	Interface	T1 + 60 ns	0 → CIREQ/
T3	D = Delay determined by ACU-controller priority	T2 + D	0 → GAITAD/

TIME	DEFINED BY	TIMING RELATIONSHIP	RTL
T4	Interface (≤ T3 + 30 ns)	T3 + 20 ns	INTDAT (8) → IODAT(8)
T5	Interface (≤ T3 + 60 ns)	T3 + 30 ns	0 → INTEN/
T6	S = Set Time determined by ACU	T3 + S	1 → GAITAD/
T7	Interface	T6 + 10 ns	0 → INT DAT(8)
T8	Interface	T6 + 30 ns	1 → INTEN
T9	Interface	T6 + 40 ns	1 → CIREQ/
T10	Controller	Reset by Read Result Type Command	1 → INT/

Configuration Code RTL Description.

TIME	DEFINED BY	TIMING RELATIONSHIP	RTL
T1	MTC Input instruction	Activated By SEL instruction	0 → INOUT Func (3) → Func Select Register 1A 1 → DEVEN
T2	Interface	T1 + 30 ns	0 → CCREQ/
T3	Interface	T2 + 10 ns	CCDAT (8) → IO Data Register^5
T4	MTC	T1 + 200 ns	0 → STROBO/ 1 → DATIEN
T5	Interface	T4 + 10 ns	IO Data Register 5 → IODAT(8)

Echo Check Output RTL Description.

TIME	DEFINED BY	TIMING RELATIONSHIP	RTL
T1	MTC Output Instruction	Activated by SEL instruction	1 → INOUT Func (3) → Func Select Register 1A 1 → DEVEN
T2	Interface	T1 + 20 ns	1 → ECKO
T3	Interface	T2 + 10 ns	ECKO (8) → IO Data Register^5
T4	MTC	T3 + 170 ns	0 → STROBO/

TIME	DEFINED BY	TIMING RELATIONSHIP	RTL
T5	Interface	T4 + 20 ns	1 → DATOEN IO Data Register 5 → DAT/(8)
T6	Interface-D is a 200 ns delay of the ECKO signal which will clock the ECKO buffer	T2 + D	DAT (8) → ECKO/(8)
T7		ACU will time-out	

Echo Check Input RTL Description

TIME	DEFINED BY	TIMING RELATIONSHIP	RTL
T1	MTC Input instruction	Activated by SEL instruction	0 → INOUT Func (3) → Func Select Register 1A 1 → DEVEN
T2	Interface	T1 + 40 ns	1 → ECKI
T3	MTC	T2 + 160 ns	0 → STROBO/
T4	Interface	T3 + 20 ns	1 → DATIEN
T5	Interface	T4 + 30 ns	ECKI (8) → DAT/(8)
T6	Interface	T5 + 30 ns	DAT/(8) → IODAT(8)
T7		ACU will time-out	

Summary

Each major design section has now been developed using register transfer language and combined into a composite design. The overall data flow and control for the symbolic design has been fully documented in this chapter. Hence, the symbolic design phase is complete. With the symbolic design phase completed, the next phase of physically realizing the design can start. The realization and implementation of the design is the subject of the next chapter.

VI. REALIZATION AND IMPLEMENTATION

Introduction

The research and design effort represented by the previous discussions is but a part of the total effort expended to implement the design. The development of the software, hardware logic, experimentation, construction and testing are yet to be accomplished. This chapter records these final areas which are essential to realize and implement the design objectives. Chapter V clearly identified when and what the interface design must do. The remaining major design decision is how to implement the design in hardware. This chapter explains the operation of the hardware design and discusses many of the realization tasks that were performed.

System Logic

To minimize noise, increase maintenance efficiency, and fulfill design specifications, a wire-wrap printed circuit (PC) board was developed using the APPLICON Computer Aided Graphics system. The significant aspect of developing this printed circuit board is that it can be directly connected to the MTC bus structure. The APPLICON system was used to produce the necessary artwork for the production of the PC board.

The wire-wrap board developed using the APPLICON Computer Aided Graphics system is shown in Figures 6-1 and 6-2 and will be referred to as the MTC Interface Board. Figure 6-1 shows the top surface of the printed circuit board while Figure 6-2 shows the bottom surface.

A standard wire-wrap printed circuit board made by Augat was used for the second essential logic board. This board will be referred to as the Tx/Rx Logic Board. The special enclosure constructed to house this board and the two controller boards is shown in Figure 6-3.

Logic Identification and Location. Any IC designator which has an "I" suffix means that it can be found on the MTC Interface Board while any designator starting with a "C" means it can be found on the Tx/Rx Logic Board. An IC designator which has an "X" or "Y" suffix indicates it is a terminal connection on the MTC Interface Board or Tx/Rx Logic Board, respectively.

The IC component layout for the MTC Interface Board is shown in Figure 6-4. The dot by each IC package identifies pin number one. Figure 6-5 shows the IC component layout for the Tx/Rx Logic Board. The notation on these boards was used to build the wiring lists. Information concerning the wiring lists for each of the boards can be found in Appendix J. A cross-reference list for identifying the exact IC type is located in Appendix H.

The internal logic for the interface design is as shown on the schematics in Appendix F. The notation adjacent to signal mnemonics has the following meaning:

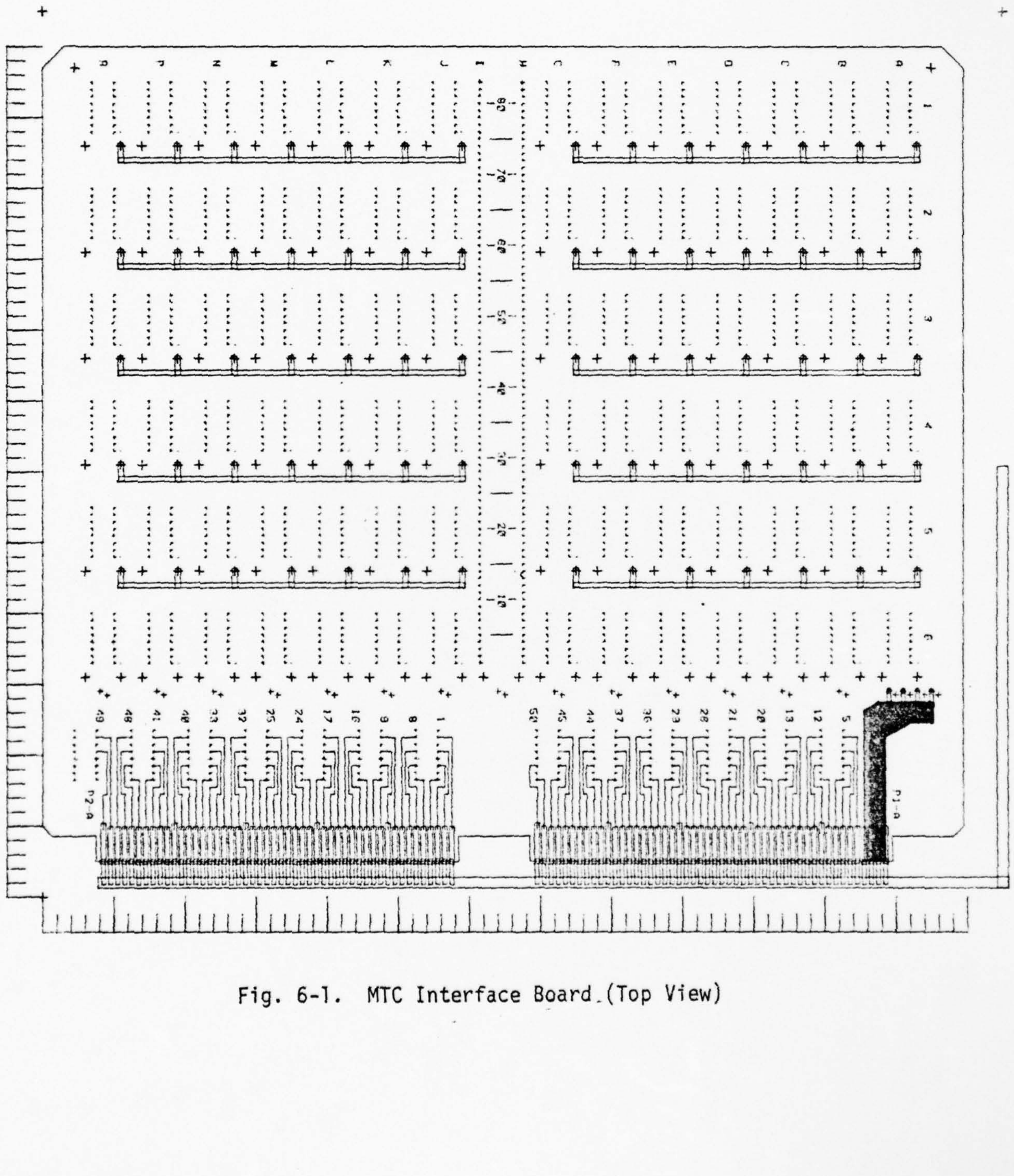


Fig. 6-1. MTC Interface Board (Top View)

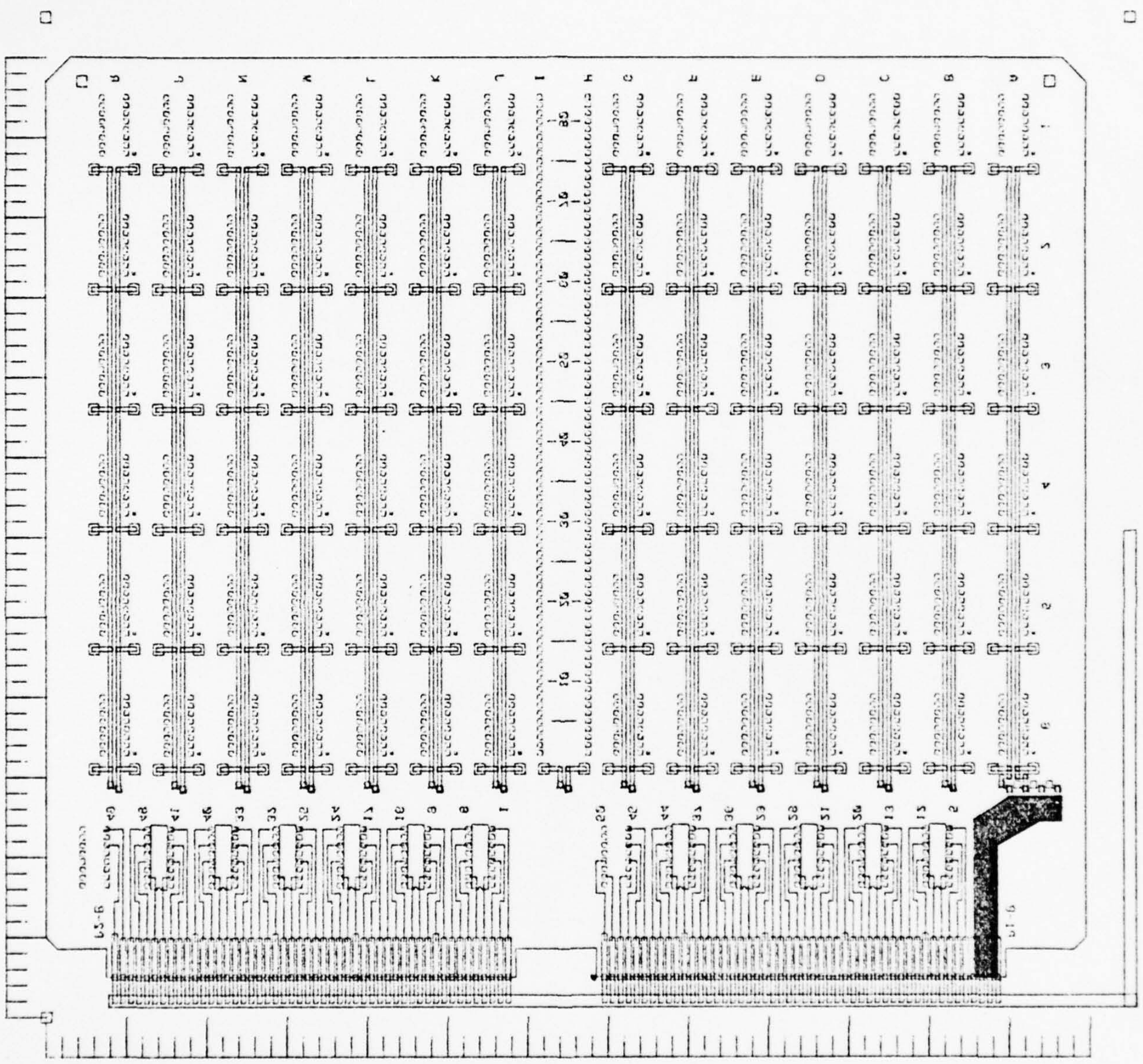
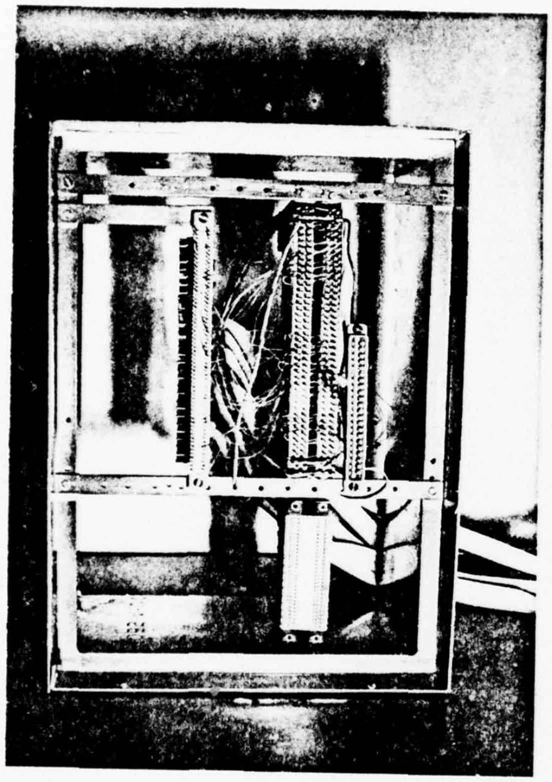
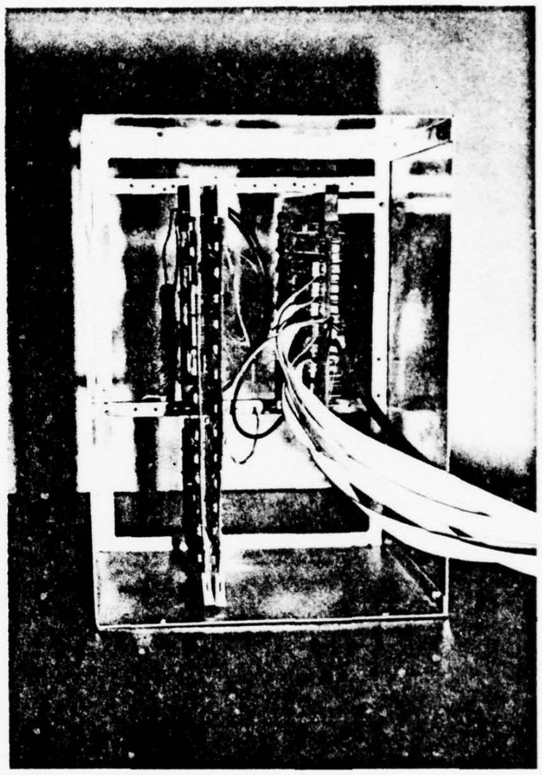


Fig. 6-2. MTC Interface Board (Bottom View)



(FRONT VIEW)



(REAR VIEW)

Fig. 6-3. Disk Controller Enclosure

Scale: 1/2" = 1"

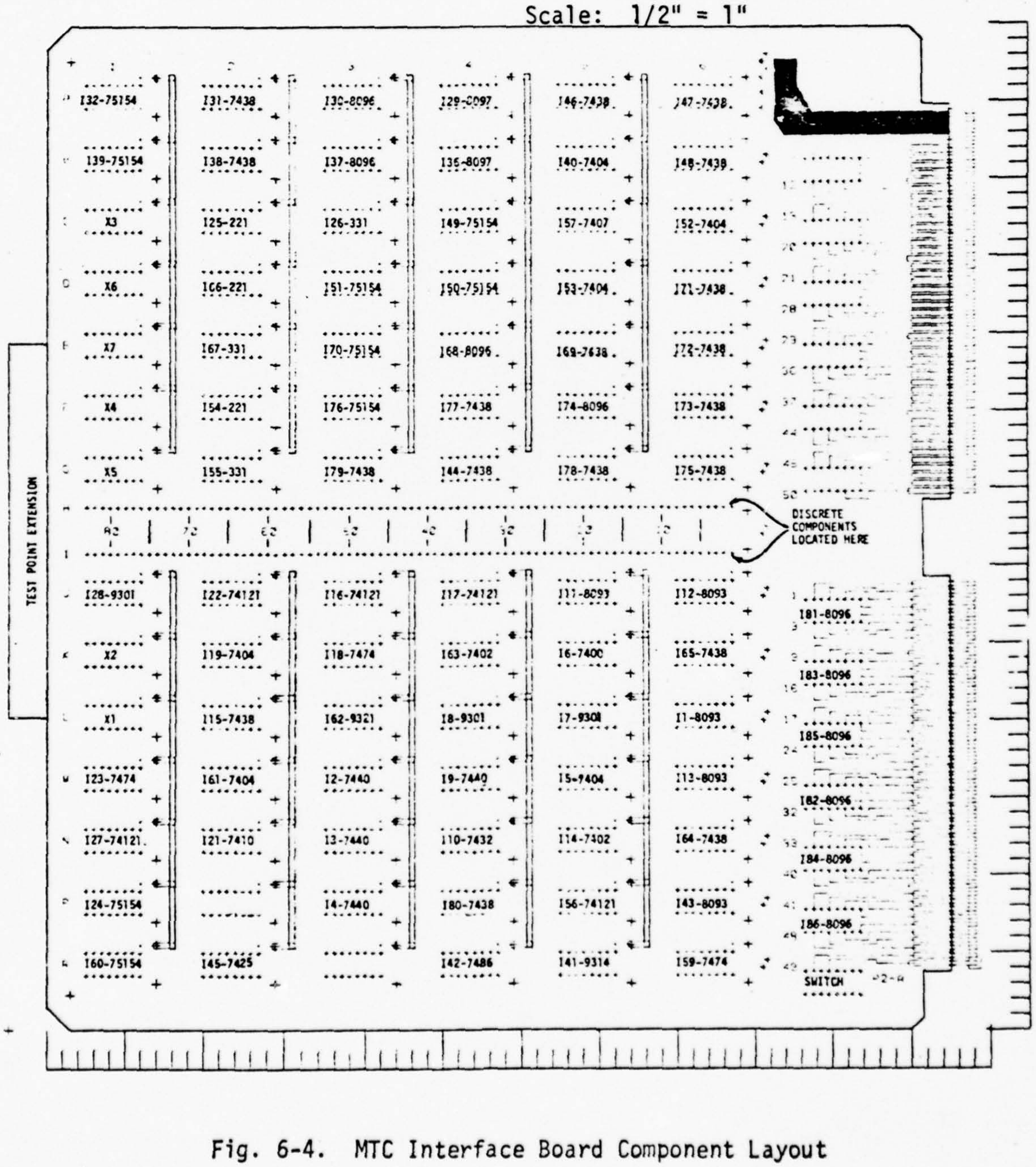


Fig. 6-4. MTC Interface Board Component Layout

Component Side

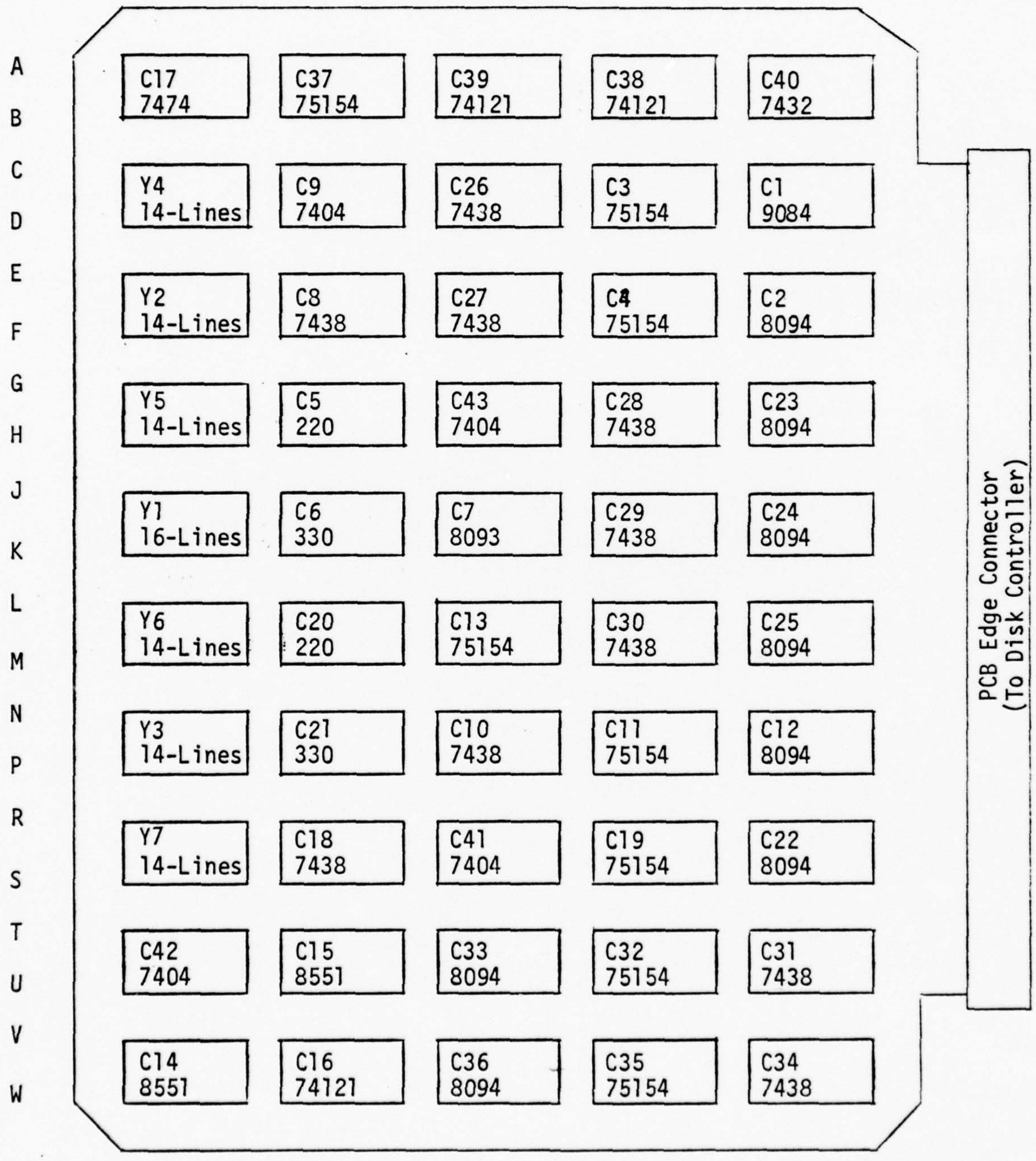
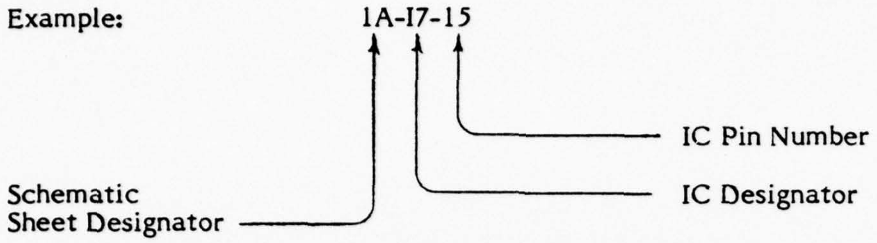
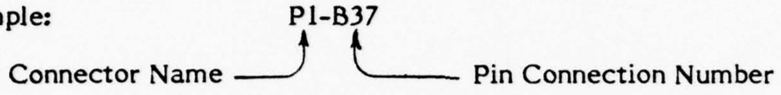


Fig. 6-5. Tx/Rx Logic Board Component Layout

Example:



Example:



Internal System Logic. Using the results of the symbolic design and register transfer language of Chapter V, the internal logic was implemented as shown in the schematics of Appendix F. A detailed design analysis of the operation of this logic can be found under the Hardware Logic section.

Transmission Facilities

Line Driving and Receiving. The main design consideration for the transmission lines is that they provide good quality signals as noted in the specifications. Either twisted pair or coaxial cable laid for minimum crosstalk is recommended for long cable lengths and for applications requiring extreme physical durability of the cable. For this application, a 10 foot separation between the Floppy Disk Drive and the MTC minicomputer was considered adequate. Therefore, a flat ribbon cable assembly with alternating signals and grounds was used for the transmission lines.

The characteristic impedance of rib cable is 120 ohms $\pm$18 ohms. The lines are terminated in their characteristic impedance to eliminate unacceptable reflections (ringing). The values of the resistors for the resistive network were also determined by the fact that resistive pull–ups at the receiving end increase noise margins.

Line drivers (SN 7438 and SN7440) and line receivers (SN75154) were employed for additional noise immunity and to insure good quality signals. The transmission line driving and receiving circuits used for this design are shown in Figure 6-6.

PC Board Interconnections. IC circuit interconnections are important because these interconnections also function as transmission lines. Since open wire connections between TTL circuits should not be bundled, tied, or routed together; the point-to-point wire-wrapping technique was used, above a ground plan which reduces coupling between conductors.

Unused Gates

Unused gate inputs were tied to ground or to V_{CC} through high pull–up resistors (current limiting resistors of ~1K ohms) to produce high outputs as necessary. This was done because it lowers the power dissipation and supplies a logic high at the gate output which can be used at unused inputs to other gates.

Unused Inputs

Theoretically, an unconnected input assumes the high logic level but practically speaking it is in an undefined logic state because it tends to act as an antenna for noise. Only a few 100 mV of noise can cause unconnected inputs to go to the logic low state (Ref 16:15-11). It is poor design practice to leave unused inputs floating. Hence, unused inputs were tied logically low or high while insuring proper output operation.

Decoupling

Decoupling capacitors are on each PC card power bus and at every TTL package. RF disk cpacitors of .01 uF are used at each TTL package while 33 uF capacitors are used for the power busses.

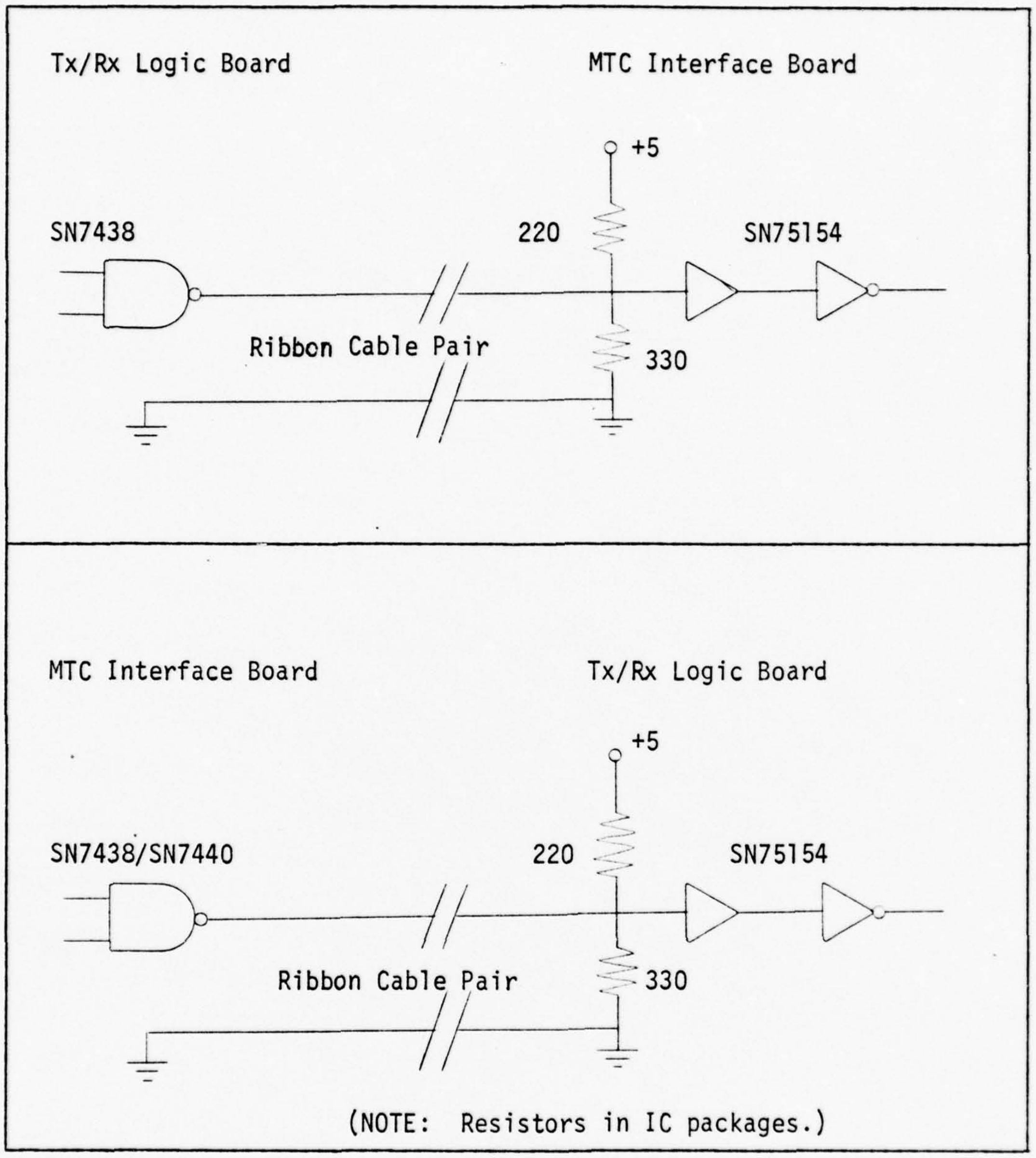


Fig. 6-6. Driver/Receiver Circuits

Hardware Logic

Most of the subheadings under Hardware Logic refer to one of the blocks in Figure 5-3. Therefore, the reader may find it helpful to refer to Figure 5-3 while reading the Hardware Logic section.

Function Tx/Rx. The Function Tx/Rx logic (Block 1) contains driver logic (SN7440) and receiver logic (SN75154) to insure high quality transmission line signals. Tri-state buffers (DM8093 and DM8094) are used to interconnect the MTC and DOS systems with the interface to insure logic compatibility. The transmission lines are terminated with a resistive network equal to the characteristic impedance of the tramsission line (approximately 120 ohms). (see Figure F-1).

Function Select Enable. The interface logic must determine whether it is supposed to perform an operation or if the FDD controller is supposed to perform the function. This action is accomplished by using 1-OF-10 decoders and appropriate gating logic as shown in Figure F-2.

Configuration Code Select. The Function Select Enable (Block 1A) uses 1-OF10 decoders to determine the function to be performed. When an input function of seven is detected, the configuration code is returned on the IODAT lines. The fixed configuration code of $2D_{16}$ is asserted when the CCREQ/ signal is applied to the tri-state buffers as shown in Figure F-2.

Delay Logic. The interface design has four critical delay logic networks. Each of the delays are generated using a monostable multivibrator (SN74121) and a D-type flip-flop (SN7474). As an example, the DEL11 signal of Figure F-4 will be explained.

The signal to be delayed, DEL1I, (active-low) enters the monostable multivibrator (one shot) which produces a negative pulse. The pulse width duration is determined by the resistor-capacitor (RC) network. The following formula and the characteristic graphs of the SN74121 (found in the TTL Data Catalog of Texas Instruments, Inc.) were used to calculate the resistor and capacitor values for the desired delays (50, 100, 200 nsec): (e.g.)

$$t = R_T\, C_T\, \mathrm{Log}_e\, 2$$

$$50 = R_T\,(10\ {}_pF)\,(.69)$$

$$R_T = 7.2\text{K ohms}$$

The positive going edge of the negative pulse is used to clock the D-type flip-flop (SN7474). Hence, the signal is delayed by the amount equal to the pulse width duration. When the delayed input signal returns to its inactive high condition, the signal is inverted by an inverter gate (SN7404) which resets the D-type flip-flop. The delay network is then ready for the next activation.

The same delay action network is used for all the required delays in the interface design.

Input and Output Control. The MTC utilizes two signals acting together, INOUT and STROBO/, to inform peripheral devices of either an input or output instruction. The disk controller, on the other hand, utilizes two separate and distinct signals, IORC/ and IOWC/. The design logic to convert the two MTC control signals to the two required disk controller signals is straight forward as shown in Figure F-3. The important note, however, is the necessary timing considerations. Since the STROBO/ signal is used to derive the IORC/ or IOWC/ signals and activate the address lines, it is critical that the IORC/ or IOWC/ signal is not applied until after the address lines have stabilized. This is accomplished by utilizing the delay network shown in Figure F-4.

Transfer Acknowledge. Figure F-6 represents the logic required to perform the "hand-shaking" operations between the MTC and DOS systems. The circuitry is complicated by the fact that the XACK/ signal is used for both DMA operations and I/O operations. The XACK/ signal is transferred through as the STROBI/ signal to provide I/O status. A critical requirement, as noted in the timing diagram of Figure 5-11, is that the XACK/ signal must be delayed by approximately 100 nsec in response to an input instruction. The delay is generated as discussed under the Delay Logic subheading (see also Figure F-7).

While the STROBI/ signal is used for I/O operations, the MEMRSM/ (XACKEN/) signal is transformed into the XACK/ signal for DMA operations. Since the XACK/ line is bi-directional, tri-state logic is used to insure only one signal can be asserted at a time.

Device Selection. The device select logic (Block 7) recognizes and decodes the I/O SEL command. The device select logic includes an eight position switch (only three positions used) for base address assignment, three SN7486 EXCLUSIVE-OR gates for address recognition, a four-bit latch for device capture and assorted gating circuits as shown in Figure F-13.

Recall from Chapter III, that the MTC ACU specifies device selection by executing the SEL instruction. The three I/O data bits (IOD1-IOD3) select the DOS if they match the base address that is assigned by setting three positions of switch S1A. These three switch positions each feed one input on three EXCLUSIVE-OR gates. If IOD1-IOD3 match the switch-selected base address, the SN7425 NOR gate (I45-12) is activated and, in turn, enables the address lines (ADRO/-ADRF/).

Address Control. The Memory Address Control logic, (Blocks 8 and 9) like the I/O Data logic, insures system compatibility through the use of line drivers (SN7438) and line receivers (SN75154). Open-collector 2-input NAND buffers are used to drive the address lines. One input of the NAND buffer being used for control purposes. The one-to-one conversion of the first 14 address lines between the MTC and DOS systems is shown in Figures F-14 and F-15.

DMA Logic and Control. The disk controller can either read or write data in a DMA operation. There is a significant difference, however, between the disk operating system and the MTC. Namely, the DOS is an 8-bit system while the MTC is a 32-bit system. The interface design takes into account this significant difference.

The reader should reference the schematics during this discussion. For this design, address bits E and F determine which 8-bit byte of the four MTC memory bytes will be read. If address bits E and F are both in the inactive high state, then byte 0 of the 32-bit word is read. Since all 32-bits (four 8-bit bytes) of information are asserted during a read instruction, the logic is designed to allow any one of the four bytes to be read. This action is accomplished thru the use of a 1-OF-4 decoder and control signal gates (RBYTEN signal) as shown in Figure F-16.

The MTC normally has the capability to write into any one or all four of the memory word bytes. However, since the DOS is an 8-bit machine, only one byte of information will be written at a time. The circuit action is basically the same as that for a read operation. The difference is in the control signals which are applied and the replication of the byte of information to be written. The DOS applies the 8-bit byte to each of the four bytes while the ACU decodes the WRTBYT lines to determine which byte of the memory word will actually get the information.

The MRDC/ and MWTC/ signals are used to request DMA (PTAREQ/) because when these signals are activated low all the information required by the MTC is available. The request for DMA (PTAREQ/) remains active low until the ACU verifies the availability of the DMA channel (PTARSV/). When the PTARSV/ signal goes active low, the ACU's DMA timing sequence is activated. Once the DMA timing sequence is activated, it is important that the PTAREQ/ signal become inactive high so that this signal will not be interpreted again as a request for DMA before the current request has completed. To insure that this does not occur, a D-type flip-flop is used. It is set by a MRDC/ or MWTC/ control signal and then subsequently reset when the ACU activates the PTARSV/ signal.

REDGATE, MDAT, and MEMRSM/ are automatically asserted by the ACU through its timing sequence.

Another important consideration is to note that the address and WRTBYT/ signals must be activiated by the PTARSV/ signal and deactivated immediately after the MEMRSM/ signal is activated low. This action is accomplished utilizing a D-type flip-flop and a monostable multivibrator. The D-type flip-flop is used to store and activate the address and WRTBYT lines while the monostable multivibrator is used to generate a pulse which will reset the D-type flip-flop there by deactivating the address and WRTBYT/ lines. (see Figure F-16 in Appendix F).

Interrupt Control. After completing the operations of a command instruction, the disk controller microprogram sets its interrupt flip–flop. Setting the interrupt flip-flop is the same as an interrupt request which the ACU recognizes. The ACU then follows with a READ RESULT TYPE command which checks the completion status of the prior transaction. It is the READ RESULT TYPE command which resets the interrupt flip–flop to allow subsequent commands.

Interrupt requests are asserted on the CIREQ/ line. This action is accomplished using a D-type flip-flop (SN7474). When the CIREQ/Line is pulled low the MTC ACU responds by asserting GAITAD/. When the ACU asserts the GAITAD/ signal, the controller reponds within 60 nsec by asserting its address on the IODAT lines. This information on the IODAT lines informs the ACU of which device or devices is requesting the interrupt. ACU priority logic determines which interrupting device and channel is serviced first since all channels may interrupt simultaneously.

After the channel has been serviced by the appropriate MTC ACU program, the GAITAD/ line is deactivated which in turn deactivates the CIREQ/ control line. A one shot is used to accomplish this function. This action enables the interrupt logic to perform the next interrupt request. The priority of the FDD channel is shown in Figure 6-7. The above action is accomplished using a monostable multivibrator, D-type flip-flop, 1–OF–10 decoder, and peripheral logic as shown in Figure F-8.

MTC CHANNEL	DEVICE	PRIORITY	INTERRUPT ADDRESS
0	Control Panel	1	0
1	Comm Channel	2	1
2	Card Reader	3	2
3	FDD	4	3
4	NONE	5	4
5	NONE	6	5
6	NONE	7	6
7	NONE	8	7

Fig. 6-7. MTC/FDD Channel Priority Settings

The channel number of the disk controller is determined by address bits 1-3. Recall that these three bits are variable and must be set on the interface card of the disk controller. These three bits are decoded by a 1-OF-10 decoder which asserts one of the eight IODAT lines low. In addition this data is only asserted on the IODAT lines when the INT/ and GAITAD/ control signals are asserted low. This control is necessary because the IODAT lines are used for more than one function. Hence, this insures that interrupt data will be asserted at the proper time.

I/O Data Select Logic. As mentioned earlier, the IODAT lines serve several functions. The result is a complex network of switching and control logic as shown in Figures F-9 and F-12. To insure signal compatibility between the systems and the interface and to insure good quality transmission signals between the devices, the I/O Data Select logic is implemented using tri-state buffers (DM8094, DM8096, and DM8097), line drivers (SN7438), and line receivers (SN75154), along with appropriate gating logic for the control signals.

IOCLK VS BCLK and CCLK. In order for the disk controller to function properly, a 10 MHz clock (BCLK and CCLK) is required. The MTC I/O bus provides a 5 MHz clock (IOCLK). In order to supply the controller with the required 10 MHz clock frequency from the 5 MHz I/O clock, two monostable multivibrators are employed. One of the monostable multivibrators is activated on the positive going edge of the IOCLK while the other is activated on the negative edge of the IOCLK. The resulting outputs of the monostable multivibrators are OR'ed together to produce the desired 10 MHz clock.

The following formula and the characteristic graphs of the SN74121 (found in the TTL Data Catalog of Texas Instruments, Inc.) were used to calculate the resistor and capacitor values:

$$t = R_T\, C_T\, \mathrm{Log}_e\, 2$$

$$50 \text{ nsec} = R_T\, (10\ _pF)\, (.6931471806)$$

$$R_T = 7.2\text{K ohms}$$

Due to component tolerances, variable 10K ohm resistors were used so symmetrical pulsewidths (50 nsec) could be produced.

Power Supply

The power requirements for each of the boards was not meticulously calculated due to the fact that the MTC has a 5 VDC, 100 amp power supply. It is estimated that the PC boards located within the special disk controller enclosure require a maximum of 10 amps. Therefore, a power cable harness capable of handling such a load was carefully selected.

System Interconnection

The final system interconnection is illustrated in Figure 6-8. P1 is the 100-pin connector which is connected to the MTC backplane. P2 is a 122-pin connector that holds the general purpose wire-wrap board called the Tx/Rx Logic Board. P3 and P4 are 86-pin connectors which hold the Channel and Interface Boards, respectively. P5 is a 44-pin connector which holds the special (wire-wrap) test board. The Test board must be removed when the system is in operation. X1-X7 identifies the seven IC terminal connectors which attach to the Interface Design Board, while Y1-Y7

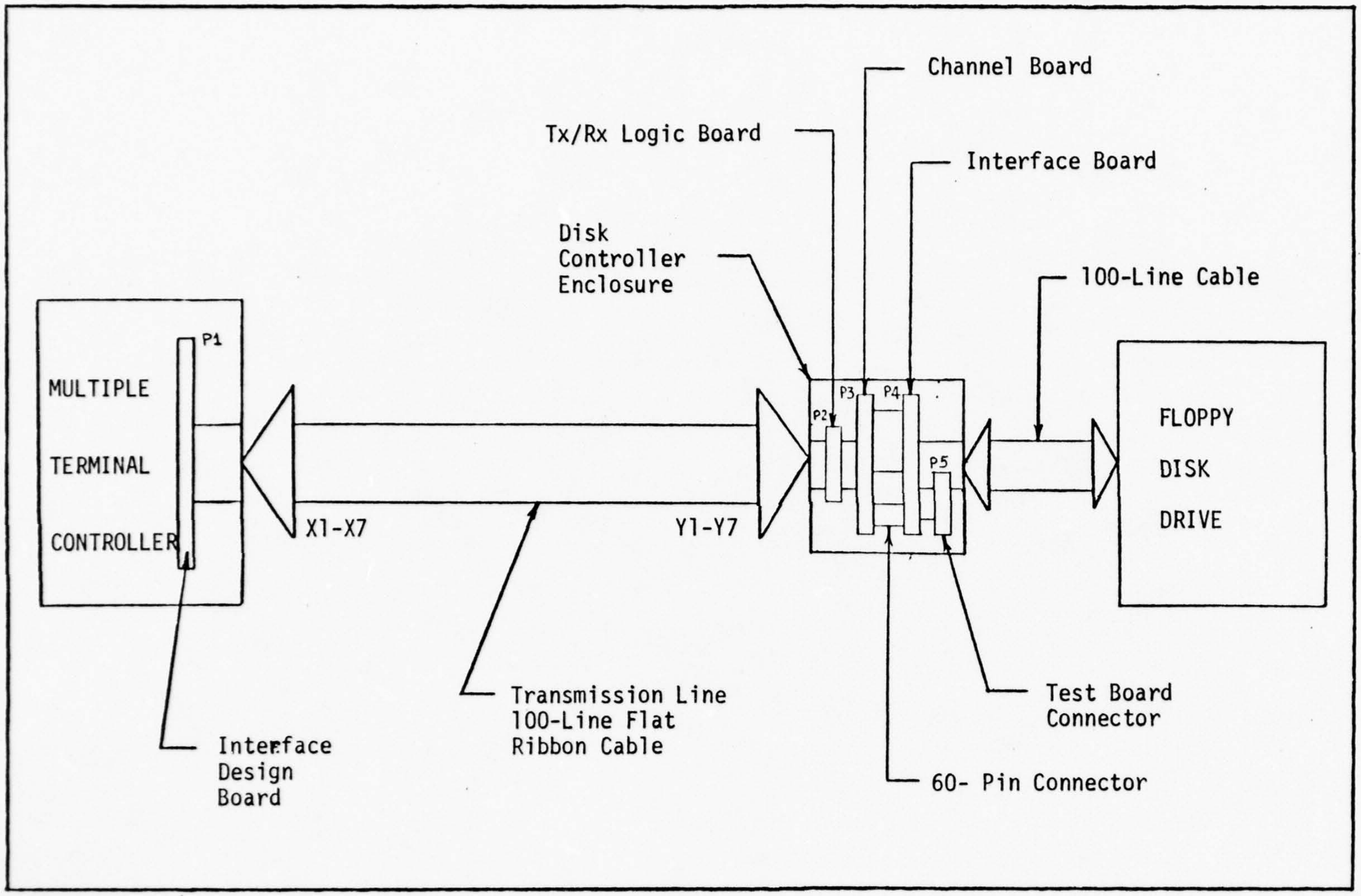


Fig. 6-8. Final System Interconnections

identifies the seven IC terminal connectors which attach to the Tx/Rx Logic Board. The Disk Controller Enclosure was specially designed and constructed for this project (see also Figure 6-3).

Construction and Testing

General. The design was partitioned into seven construction/testing phases as follows:

(1) Clock Generator

(2) Device Select

(3) Input/Output Control

(4) Configuration Code

(5) Echo Check

(6) Direct-Memory Access

(7) Interrupt

Once each phase was wire-wrapped, software was written to manually test all the functions. The complexity of the software varied in accordance with the function to be tested.

To eliminate the complex and time consuming task of generating cards for each test program, a program developed by R. W. Koontz called MTC DEBUG was used which allows direct memory manipulation and execution from a CRT display console. The MTC DEBUG program is based upon a debug package named CONSOLE which was created November 1, 1972.

Clock Generator. No software was necessary to check the operation of the clock generator. The variable resistors in the clock generator circuitry were adjusted in order to get symmetrical, 50 nsec pulse widths.

I/O Functions. A sample program written to test one of the I/O functions

(the output of 8-bits of data) is illustrated in Figure 6-9. Similar programs were written and executed to check each of the I/O functions which include configuration code and echo check. In the process of testing I/O functions, the device select logic is also tested.

DMA. In order to test the DMA logic, special test logic was designed as shown in Figures 6-10 and 6-11. The test logic will be discussed briefly. The test logic simulates memory reads and memory writes initiated by the disk controller. When simulating a memory read, the data switches must open (dot up). This allows the data to be returned and displayed by the light emitting diodes (LED's). A lit LED represents a logical Ø. The memory address from which data is read or where data is written is hardwired to location 1000_{16}. When simulating a memory write, the data switches can be set to the bit pattern desired. Switch SX-5, when open, allows cyclic memory reads or writes.

Interrupt. Finally, the interrupt logic was manually tested using part of the DMA test logic (see Figure 6-10). First, dip switches SX-7 and SX-3 were closed, leaving all other switch settings open. Then,
SX-7 was opened which simulated the INT/request.

Summary

Chapter VI has identified the hardware logic that was actually used to implement thedesign. The operation of the logic was discussed in detail, along with explaining many techniques that were employed to increase signal integrity and system efficiency. The component layout for each

MEMORY LOCATION	CONTENTS OF MEMORY LOCATION IN HEXIDECIMAL CODE	ATLAS ASSEMBLY LANGUAGE INSTRUCTION
120	6C000000	XX SEL I,0
124	6A030000	YY INP 0, F=3
128	5C000124	UTR YY
12C	4E030000	ZZ OUT 0, F=3
130	5C00012C	UTR ZZ
134	6C000030	SEL I, 30
138	4E010000	OUT 0, F=1
13C	14000000	NOP
140	5C000120	UTR XX

Fig. 6-9. Sample I/O Test Program

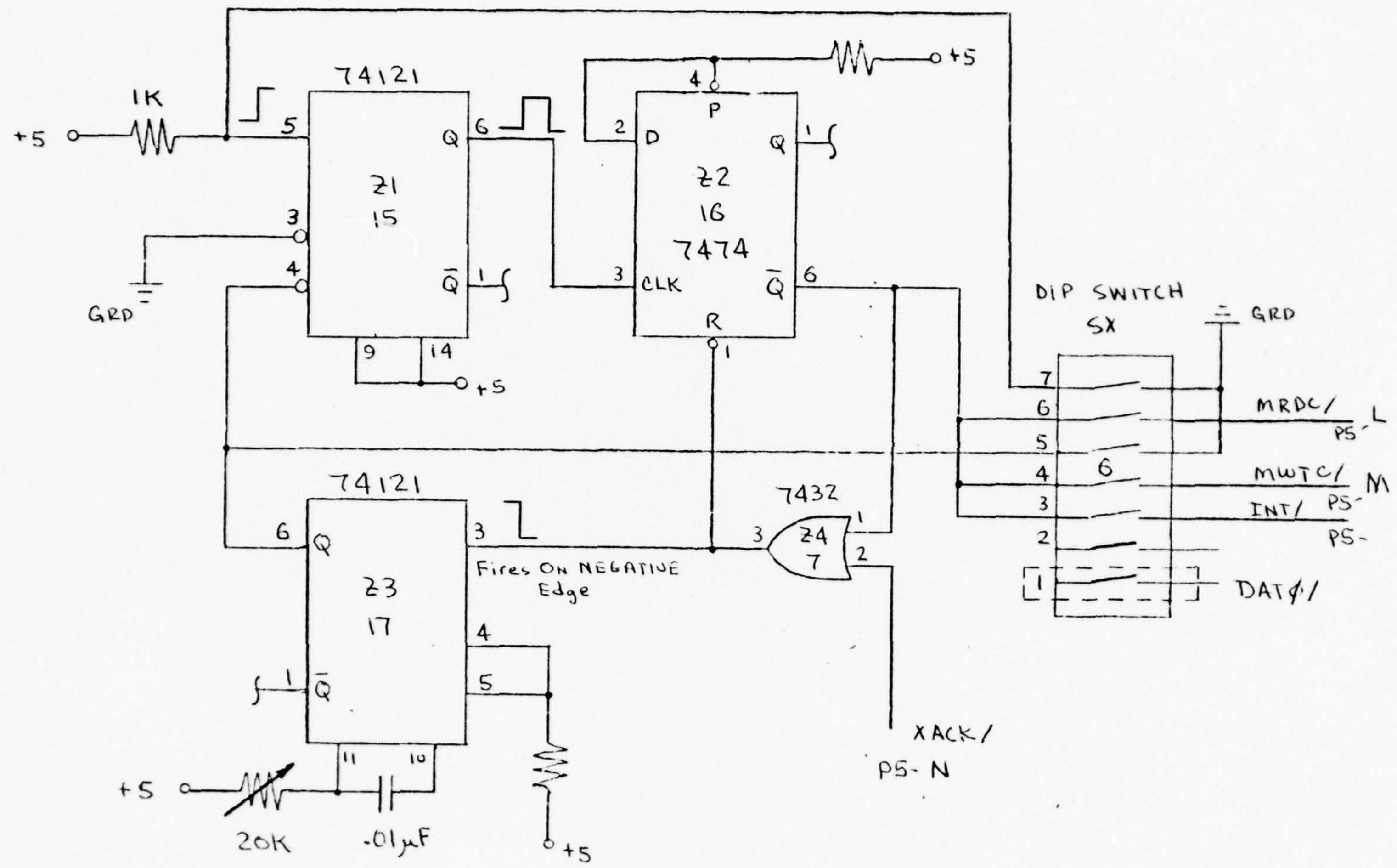


Fig. 6-10. Test Logic-Control

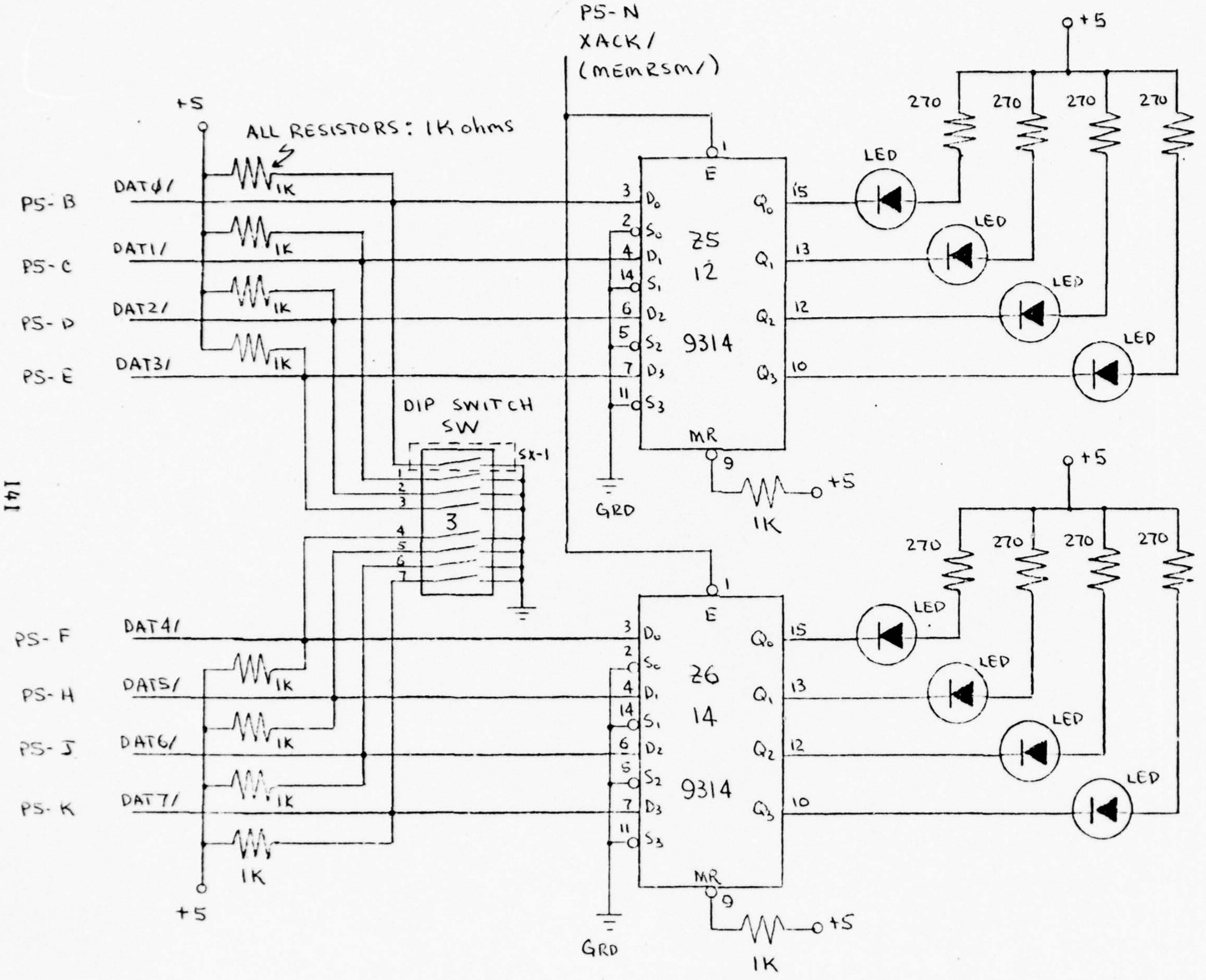


Fig. 6-11. Test Logic-Data Manipulation

board and the final system interconnections were succinctly illustrated. Finally, the seven construction and testing phases were explained while observing a sample test program and the hardware test logic.

VII. CONCLUSIONS AND RECOMMENDATIONS

Introduction

The results of testing the interface design show quite conclusively that it is indeed possible and highly feasible to expand the capabilities of the MTC system into a stand-alone system. Hopefully, the results of this project will be used to profitably convert the limited capabilities of the MTC into a fully operational stand-alone system.

Objectives

The objectives of this effort were achieved. The three general design areas of I/O control, DMA, and program interrupt were each designed, implemented, and tested to insure proper operation. The interface design was partitioned into centrally interconnected subsystems using wire-wrap PC boards which allows considerable flexibility in terms of modifications. Emphasis throughout the design was placed on accessibility which resulted in a system that the maintainer can easily investigate and probe.

Software Requirements

Much remains to be done in the software area however before the interface design and the diskette drive unit can be fully integrated into the MTC system. A first priority, of course, would be to set up the diskette drive unit and connect it to the interface design so that all critical timing parameters can be tested and adjusted as appropriate. Once this is accomplished an effort should be directed towards writing a software diagnostic package which would check the operation of the hardware logic. Finally, a software driver program must be designed and developed which will provide the stand-alone capability desired of the MTC system.

Enhancements

Some additional projects and MTC enhancements include:

(1) Design and develop the software for the floppy disk drive used in this project.

(2) Expand the I/O system to include other devices, such as a cassette tape unit, a standard magnetic tape drive, a cartridge disk, or a disk pack drive.

(3) Add a reject function capability and incorporate a 16-bit transfer instead of an 8-bit. In addition, an effort to minimize the hardware logic of this project could be asserted. For example, each of the delay networks which consist of several IC's could be replaced by a single IC.

(4) Expand the interface design to include malfunction logic which would interrupt the ACU in the event of a fault.

Recommendations

Lastly, the reader should be aware of some serious pitfalls which could totally beset the designer's project if they are not considered in the program development plan. First of all, short term projects which have hardware procurement requirements, should be defined and the hardware ordered as soon as possible in the conceptual phase; and plan on having at least three months of lead time. Secondly, in order to develop interface hardware logic, detailed schematics and drawings must be available on the system. Hence, technical data and support literature should be obtained when the system is procured. Another important consideration is the physical system interaction required on the part of the designer. If the designer must frequently, physically interface with the system, then the system must be readily available. Finally, if the project is going to take the designer through the life-cycle development phases of conception, requirements definition, design, development, testing, integration, and operation, then the designer should plan for the problems and additional delays that will invariably be encountered in the areas of documentation, logistics, configuration management, equipment failure, and personnel.

Bibliography

1. Burck, Gilbert. The Computer Age. New York: Harper-Row, 1965.

2. Cabinet Assembly. St. Paul, Minnesota: Control Data Corporation, 1974.

3. Digital Integrated Circuits. Santa Clara, California: National Semiconductor Corporation, May 1971.

4. Diskette Operating System. Santa Clara, California: Intel Corporation, October 1975.

5. Flores, Ivan. Peripheral Devices. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1973.

6. Key To Logic Smybology For Terminal Equipment. St. Paul, Minnesota: Control Data Corporation, 1975.

7. Matick, Richard E. Transmission Lines For Digital and Communication Networks. New York, New York: McGraw-Hill Book Company, 1969.

8. MDS-800 Intellec MDS. Santa Clara, California: Intel Corporation, October 1975.

9. Random Access Memory Module. St. Paul, Minnesota: Control Data Corporation, 1975.

10. Single I/O Channel, Dual I/O Channel. St. Paul, Minnesota: Control Data Corporation, 1974.

11. Synchronous Communications Channel and Input/Output Channel. St. Paul, Minnesota: Control Data Corporation, 1973.

12. Terminal Facility Controller Hardware Diagnostics System, Volume I and Volume II. St. Paul, Minnesota: Control Data Corporation, 1974.

13. Terminal Facility Controller Reference Manual. St. Paul, Minnesota: Control Data Corporation, 1974.

14. Texas Instruments Inc. Designing with TTL Integrated Circuits. New York: MCGraw-Hill Book Company, 1971.

15. The Integrated Circuits Catalog. Dallas, Texas: Texas Instruments Incormporated, undated.

16. The TTL Applications Handbook. Moutain View, California: Fairchild Corporation, August 1973.

Appendix A

Instruction List

Appendix A lists the 30 instructions of the ACU available to the programmer. Included are the mnemonic, function, description, and the immediate, direct, and indirect hexadecimal notation for the instruction. Note also that there is a listing for the operation-code field. The first listing (column 3 in the table) resulted from grouping only the bits of the op-code field for hexadecimal conversion and ignoring the rest of the 32-bit instruction word. The second, third, and fourth listings (columns 4, 5, and 6 in the table) resulted from grouping the bits of the full 32-bit word before binary-to-hexadecimal conversion of both the op-code and M fields.

Example: Determine the hexadecimal code for an LDR instruction by two methods.

First method--

Determine hexadecimal code by grouping and converting the op-code field only.

```
        00  ' 0001

             01   (see column 3 of following table)
```

Second method--

Determine hexadecimal code by grouping the bits of the full 32-bit word and, then, by converting the op-code and M fields.

```
0000 ' 0100 ' XXXX ' XXXX ' XXXX ' XXXX ' XXXX ' XXXX
```

04_{16} (see column 4 of following table)

Table IX

INSTRUCTION LISTING

FUNCTION	MNEMONIC	HEXADECIMAL CODE OP CODE (Note 1)	IMMEDIATE (Note 2)	DIRECT (Note 2)	INDIRECT (Note 2)	DESCRIPTION
Internal Data Transfer	LDR	01	04	06	05	Load Register
	LOC	02	08	0A	09	Load 1's Complement
	EXR	04	--	12	11	Exchange Register File
	LMP	06	--	1A	19	Load Multiple Precision
	STR	14	--	52	51	Store Register
	SOC	15	--	56	55	Store 1's Complement
	SMP	16	--	5A	59	Store Multiple Precision
Arithmetic	ADD	07	1C	1E	1D	Add
	AOS	08	--	22	21	Add 1 to Store
	RAD	09	--	26	25	Replace Add
	MPA	0A	--	2A	29	Multiple Precision Add
	SUB	0B	2C	2E	2D	Subtract
	SOS	0C	--	32	31	Subtract 1 from Storage
	RSB	0D	--	36	35	Replace Subtract
	MPS	0E	--	3A	39	Multiple Precision Subtract
	CPM	1D	74	76	75	Compare Magnitude
Logical	IOR	0F	3C	3E	3D	Inclusive OR
	XOR	10	40	42	41	Exclusive OR
	AND	11	44	46	45	AND
	GP-	12	--	4A	49	Generate Byte Parity, Odd or Even
Shift	SHR	05	14	--	--	Shift Register
Control Transfers	UTR	17	5C	5E	5D	Unconditional Transfer
	TOC	18	60	62	61	Transfer On Condition
	TLM	1E	78	7A	79	Transfer and Link Memory
	RTR	1F	7C	7E	7D	Return Transfer
	RPT	19	64	66	65	Repeat from Effective Address
I/O Control	SEL	1B	6C	6E	6D	Select Channel and Device
	OUT	13	4C	4E	4D	Output Transfer
	INP	1A	--	6A	69	Input Transfer
	TIN	1C	70	72	71	Test Input

Note 1: Hexadecimal code in this column was determined from the Op-code field only, grouping the bits for hexadecimal conversion as if they stood alone (XX ' XXXX).

Note 2: Hexadecimal code in this column was determined by grouping the bits of the full 32-bit word, using both the Op-code and M fields (XXXX ' XXXX ' XXXX ' XXXX ' XXXX ' XXXX ' XXXX ' XXXX). Note the difference in the grouping of the leftmost bits here as compared to that of note 1.

Appendix B

MTC Schematics

Appendix B provides the reader with the MTC schematics which played an active role in the development of the interface design. The characteristics noted in these schematics were critical to the intersystem connections. Included with the schematic diagrams are discussions of the theory of operation. Since the schematic components are in a symbolic form, the following cross-reference list is provided to aid in the understanding:

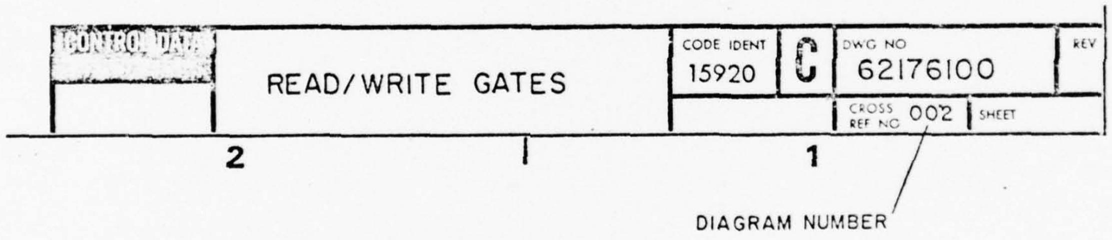
Schematic Symbolic Terminology	Corresponding Manufacturer's Part Number/Description
204 &	7438-2-Input NAND Buffer
530 MUX	8123-Tri-State Quad 2-Input Mux
164H FF	3062-Dual J-K Flip-Flop
213H &	74H11-Triple 3-Input AND
143H 1	7420-4-Input NAND
146H 1	74H04-Hex Inverter
210 &	7437-2-Input NAND
216 1	8093-Tri-State 2-Input Quad Buffer
532	9318-8-Input Encoder

DIAGRAMS

This section supplies logic diagrams with backup pages that explain circuit operation. Also, figure B-1 outlines Memory Read Operation and figure B-2 shows Memory Write Operation.

Two items identify each logic diagram as shown by the following:

CARD TYPE (See Note 1, Diagram 000.)



NOTE

Logic diagrams 62176100 describe RAM boards 2CND/2CPD with Intersil chips, and logic diagrams 62177100 describe the 2CND-1/2CPD-1 boards with Fairchild chips. Back up pages are included only with the 62176100 diagrams.

Backup Pages:

Diagram	Title	Page
000	RAM Control	5-6
001	Read/Write Gates	5-8
003	Even Bank, Bytes 0 and 1	5-12

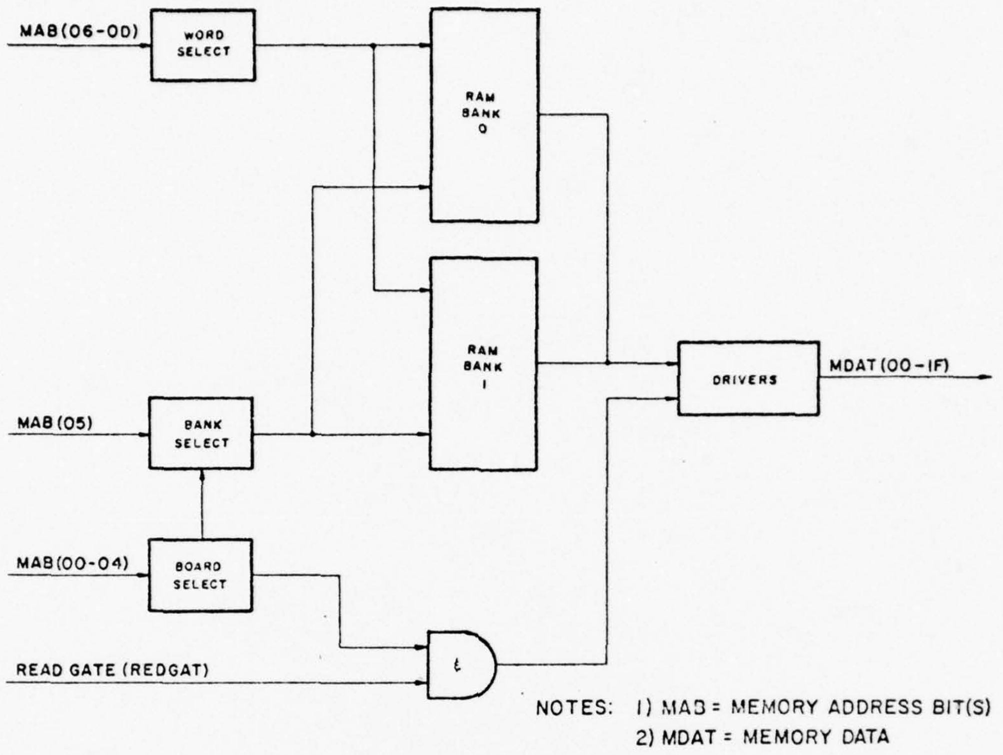


Fig. B-1. Memory Read Operation.

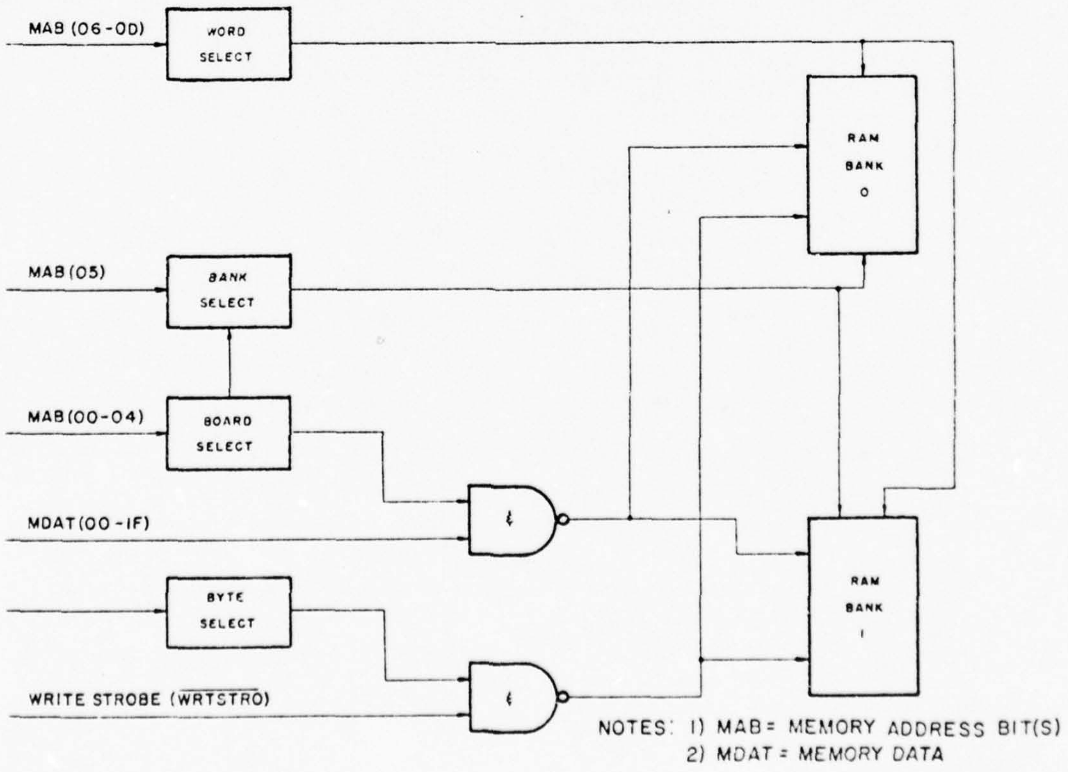
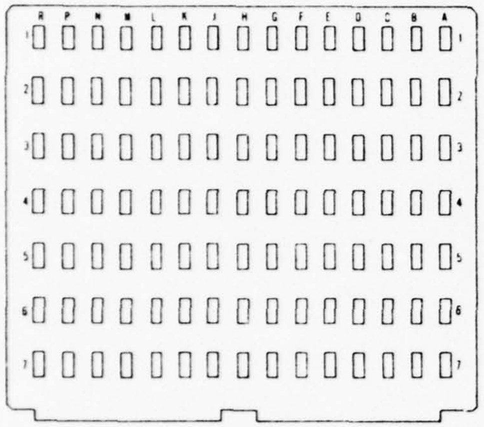


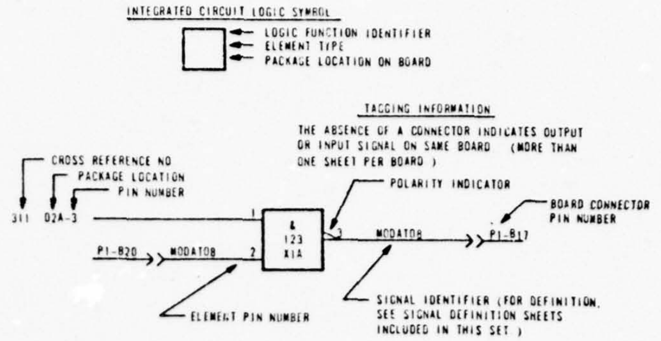
Fig. B-2. Memory Write Operation.

LOGIC DIAGRAMS

EACH PRINTED CIRCUIT BOARD IN THIS LOGIC SET UNDER IDEAL CONDITIONS WOULD BE REPRESENTED BY ONE LOGIC DIAGRAM HOWEVER DUE TO THE LARGE NUMBER OF CIRCUITS LOCATED ON EACH BOARD EXTRA DIAGRAMS ARE NORMALLY REQUIRED THE BOARD TYPE AND LOCATION OF THE BOARD REPRESENTED IS LISTED IN THE LOWER RIGHT CORNER OF THE DIAGRAM EACH SYMBOL ON THE DIAGRAM REPRESENTS A PORTION OF AN INTEGRATED CIRCUIT THE ENTIRE INTEGRATED CIRCUIT OR A DISCRETE COMPONENT EACH SYMBOL REFLECTS THE COMPONENT TYPE THE LOGIC FUNCTION PERFORMED AND THE COMPONENT LOCATION FOR INFORMATION ON THE COMPONENT OR ELEMENT TYPES USED IN THIS LOGIC SET REFER TO KEY TO LOGIC SYMBOLOGY MANUAL CDC PUBLICATION NUMBER 82172400 COMPONENT LOCATION IS DEFINED BY AN ALPHANUMERIC MATRIX DEPENDING ON THE TYPE OF PRINTED CIRCUIT BOARD USED



GENERAL LOGIC SYMBOL INFORMATION



INTEGRATED CIRCUIT LOGIC SYMBOL
LOGIC FUNCTION IDENTIFIER
ELEMENT TYPE
PACKAGE LOCATION ON BOARD
TAGGING INFORMATION
THE ABSENCE OF A CONNECTOR INDICATES OUTPUT OR INPUT SIGNAL ON SAME BOARD (MORE THAN ONE SHEET PER BOARD)
CROSS REFERENCE NO
PACKAGE LOCATION
PIN NUMBER
311 02A-3
POLARITY INDICATOR
BOARD CONNECTOR PIN NUMBER
P1-B20
MODAT08
1
2
3
&
123
X1A
MODAT08
P1-B17
ELEMENT PIN NUMBER
SIGNAL IDENTIFIER (FOR DEFINITION, SEE SIGNAL DEFINITION SHEETS INCLUDED IN THIS SET)

VOLTAGE LEVELS

1 ANALOG OR NON-LOGIC LEVELS

2 NON-STANDARD LOGIC LEVEL

3 LOGIC LEVELS

DTL LOGIC OPERATION

HIGH (LOGICAL 1) = +2 6 TO +5 0 VOLTS
LOW (LOGICAL 0) = 0 0V TO +0 45 VOLTS
SWITCHING POINT = +1 1 TO +1 9 VOLTS

TTL LOGIC OPERATION

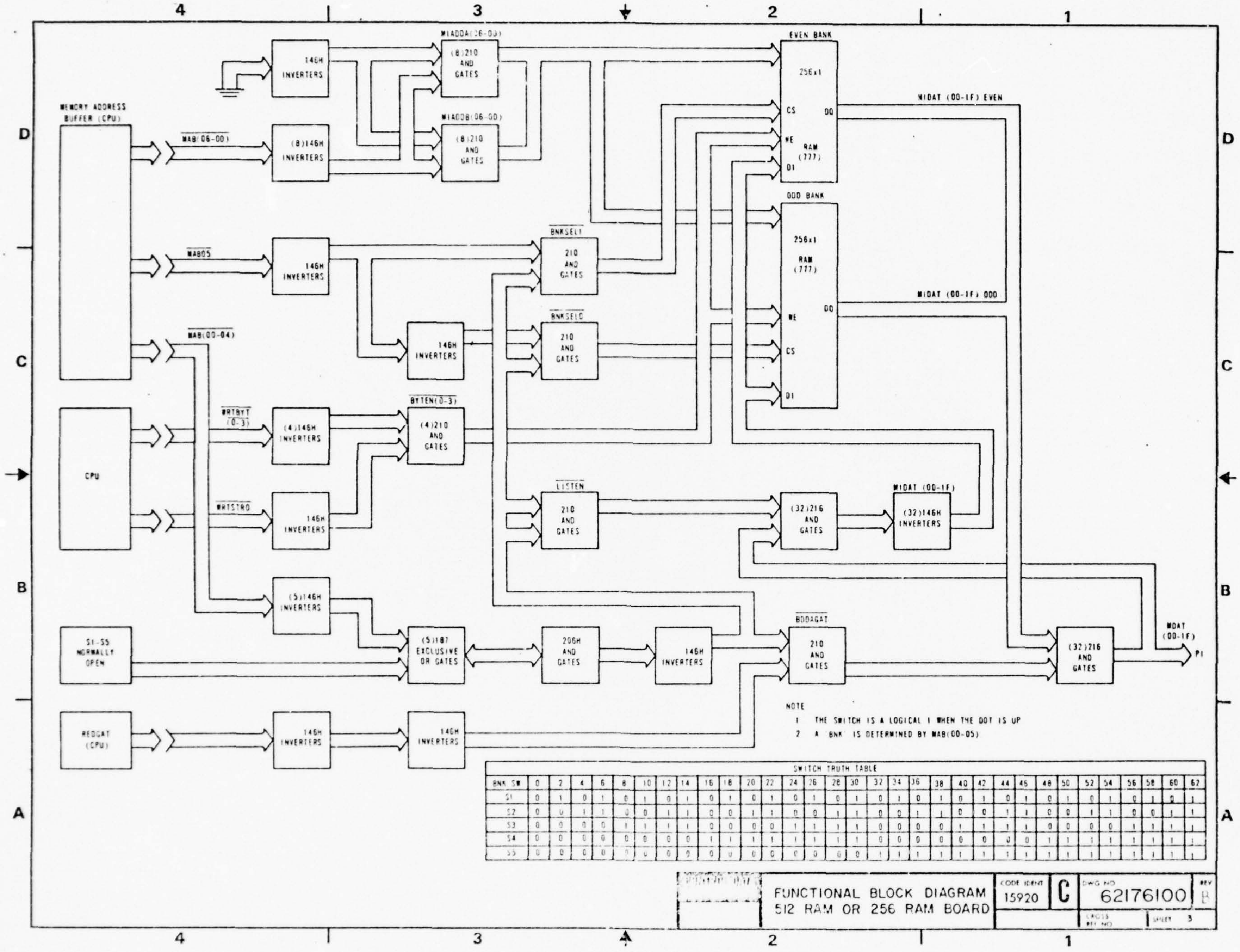
HIGH (LOGICAL 1) = +2 8 TO +5 0 VOLTS
LOW (LOGICAL 0) = 0 0V TO +0 45 VOLTS
SWITCHING POINT = + 85V TO +1 9 VOLTS

LOGIC FUNCTION IDENTIFIER

& = AND
1 = OR
1 = INVERTER
=1 = EXCLUSIVE 'OR' (ONE INPUT ONLY)
= = EXCLUSIVE 'OR' (TWO INPUTS, NO MORE, NO LESS)
⊐ = SCHMITT TRIGGER

CONTROL DATA	KEY TO SYMBOLS	CODE IDENT 15920	C	DWG NO 62176100	REV G
				CROSS REF NO	SHEET 2

Fig. B-3. Key To Symbols



NOTE
1 THE SWITCH IS A LOGICAL 1 WHEN THE DOT IS UP
2 A 'BNK' IS DETERMINED BY MAB(00-05)

SWITCH TRUTH TABLE

BNK SW	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42	44	45	48	50	52	54	56	58	60	62
S1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
S2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
S3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
S4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
S5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

FUNCTIONAL BLOCK DIAGRAM 512 RAM OR 256 RAM BOARD	CODE IDENT 15920	C	DWG NO 62176100	REV B
			CROSS REF NO	SHEET 3

62999000 B Fig. B-4. RAM Board-Functional Block Diagram

DIAGRAM 000

MEMORY ADDRESS LINES

A memory cycle starts when all of the 14 address lines are stable at the memory interface (shown here).

BOARD SELECTION is accomplished by $\overline{\text{MAB00}}$ thru $\overline{\text{MAB04}}$. When these lines exactly match the settings of address switches S1 thru S5 at location R1 or M6, the resultant low output from the Exclusive OR at location R7 will be inverted, and the high output at M7-6 will generate the $\overline{\text{LISTEN}}$ signal and enable the $\overline{\text{BODAGAT}}$, $\overline{\text{BNKSEL0}}$ and BNKSEL1 signals.

BOARD ADDRESS CROSS REFERENCE

P Register Bit	0	1	2	3	4
Memory Address Bit (MAB)	00	01	02	03	04
Diagram 000	S1	S2	S3	S4	S5
PCB Label	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}

BANK SELECTION is accomplished by $\overline{\text{MAB05}}$. When this line is a logical 0 the high output at M7-8 will generate the $\overline{\text{BNKSEL1}}$ signal at N6-11. When the Bank Selection Line is a logical 1, the high output from M7-10 will generate the $\overline{\text{BNKSEL0}}$ signal at N6-8.

WORD SELECTION is accomplished by MAB06 thru MAB0D, which are common to all banks in the total read/write memory. These bits select one of the 256 words in the selected memory bank on the selected board. The low outputs from the upper set of NAND gates are used to gate bits 0 thru F out of the selected memory bank, and the low outputs from the lower set of NAND gates are used to gate bits 10 thru 1F out of the selected bank.

WRITE STROBE (WRTSTRO)

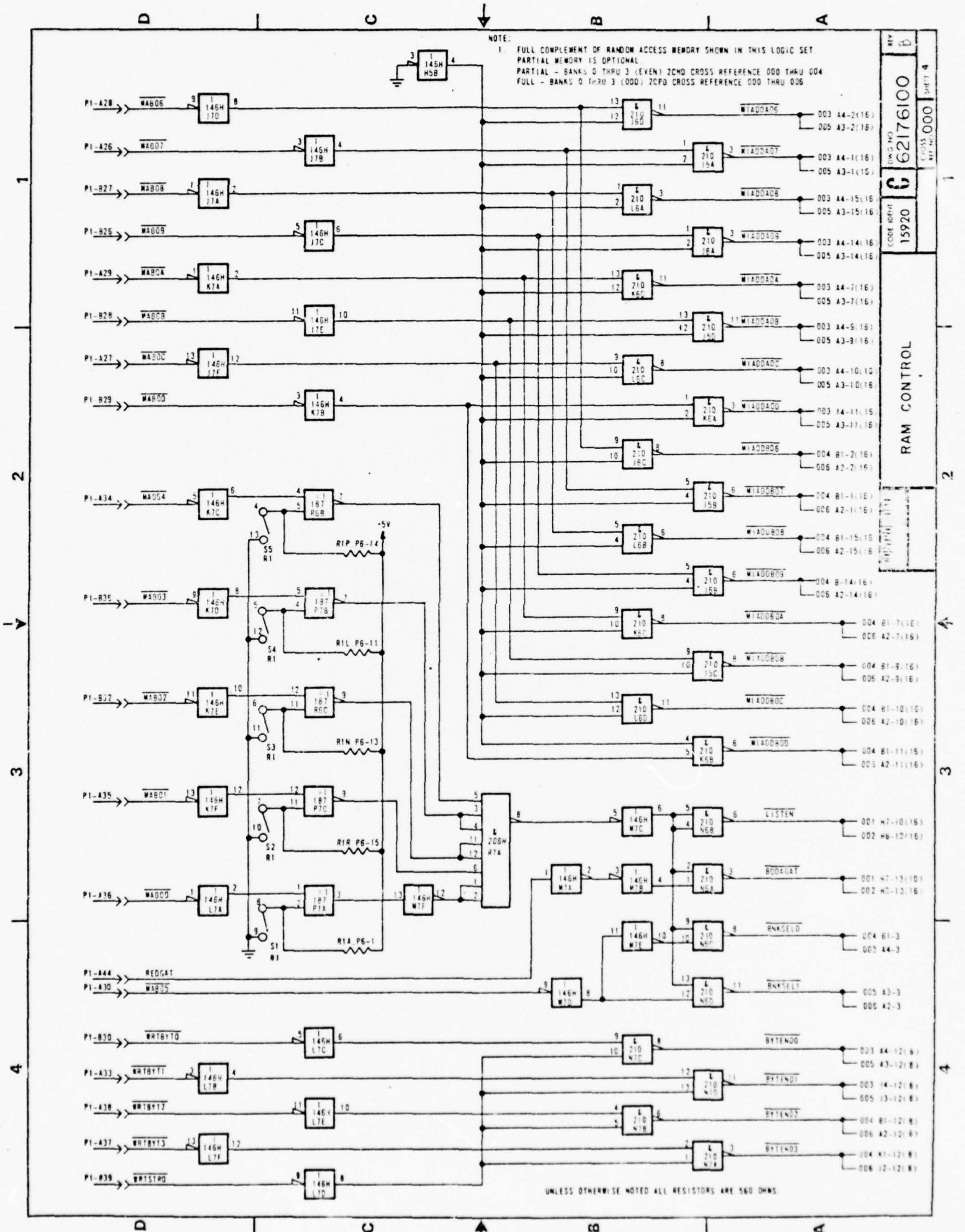
When the Write Strobe line is a logical 0, this is inverted at location L7D and the output from L7-8 is used to enable the Write Byte selection signals.

WRITE BYTE CONTROL

A logical 0 on one of the Write Byte lines will be inverted and gated through one of the NAND gates at location N7 to serve as address for the 8-bit byte to be written into memory.

READ GATE (REDGAT)

When this line is a logical 1, it generates the $\overline{\text{BODAGAT}}$ signal at N6-3, which is used to gate the memory data onto the Memory Data Lines on diagrams 001 and 002.



NOTE:
1. FULL COMPLEMENT OF RANDOM ACCESS MEMORY SHOWN IN THIS LOGIC SET
PARTIAL MEMORY IS OPTIONAL
PARTIAL - BANKS 0 THRU 3 (EVEN) 2CMD CROSS REFERENCE 000 THRU 004
FULL - BANKS 0 THRU 3 (ODD) 2CPD CROSS REFERENCE 000 THRU 006

UNLESS OTHERWISE NOTED ALL RESISTORS ARE 560 OHMS.

Fig. F-5. RAM Control

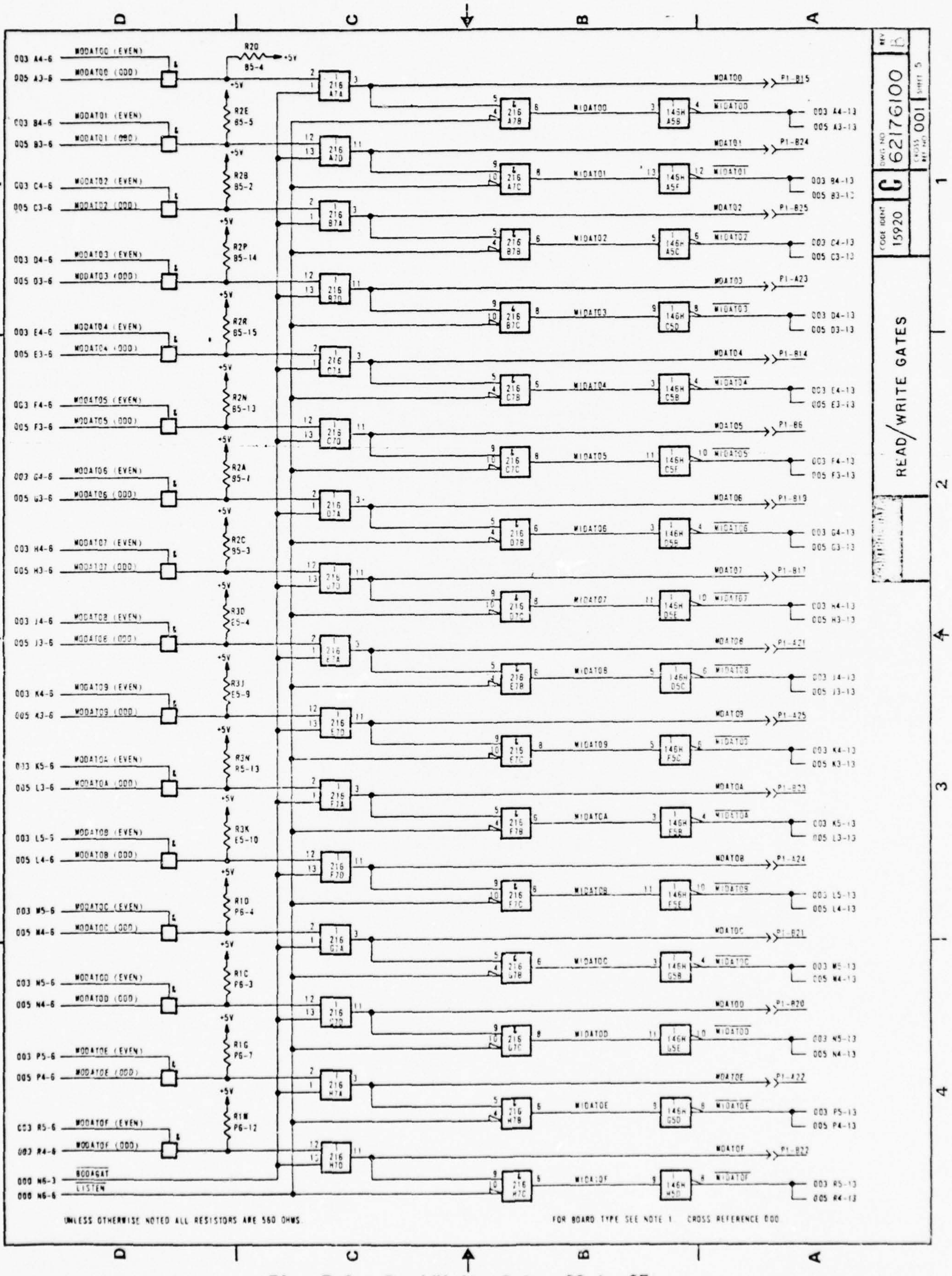
DIAGRAM 001

WRITE DATA

Data to be written in memory is applied to the board connector (P1), and the $\overline{\text{LISTEN}}$ signal gates it through a set of parallel AND gates, after which it is inverted and applied to the selected 8-bit section of the addressed word in the selected memory bank.

READ DATA

Data to be read from memory is gated onto the Memory Data Lines by the $\overline{\text{BODAGAT}}$ signal.


READ/WRITE GATES
62176100
15920
UNLESS OTHERWISE NOTED ALL RESISTORS ARE 560 OHMS.
FOR BOARD TYPE SEE NOTE 1. CROSS REFERENCE 000
MODAT00 (EVEN)
MODAT00 (ODD)
MIDAT00
BODAGAT
LISTEN

62999000 B Fig. B-6. Read/Write Gates 00 to 0F

DIAGRAM 002

See backup page for diagram 001.

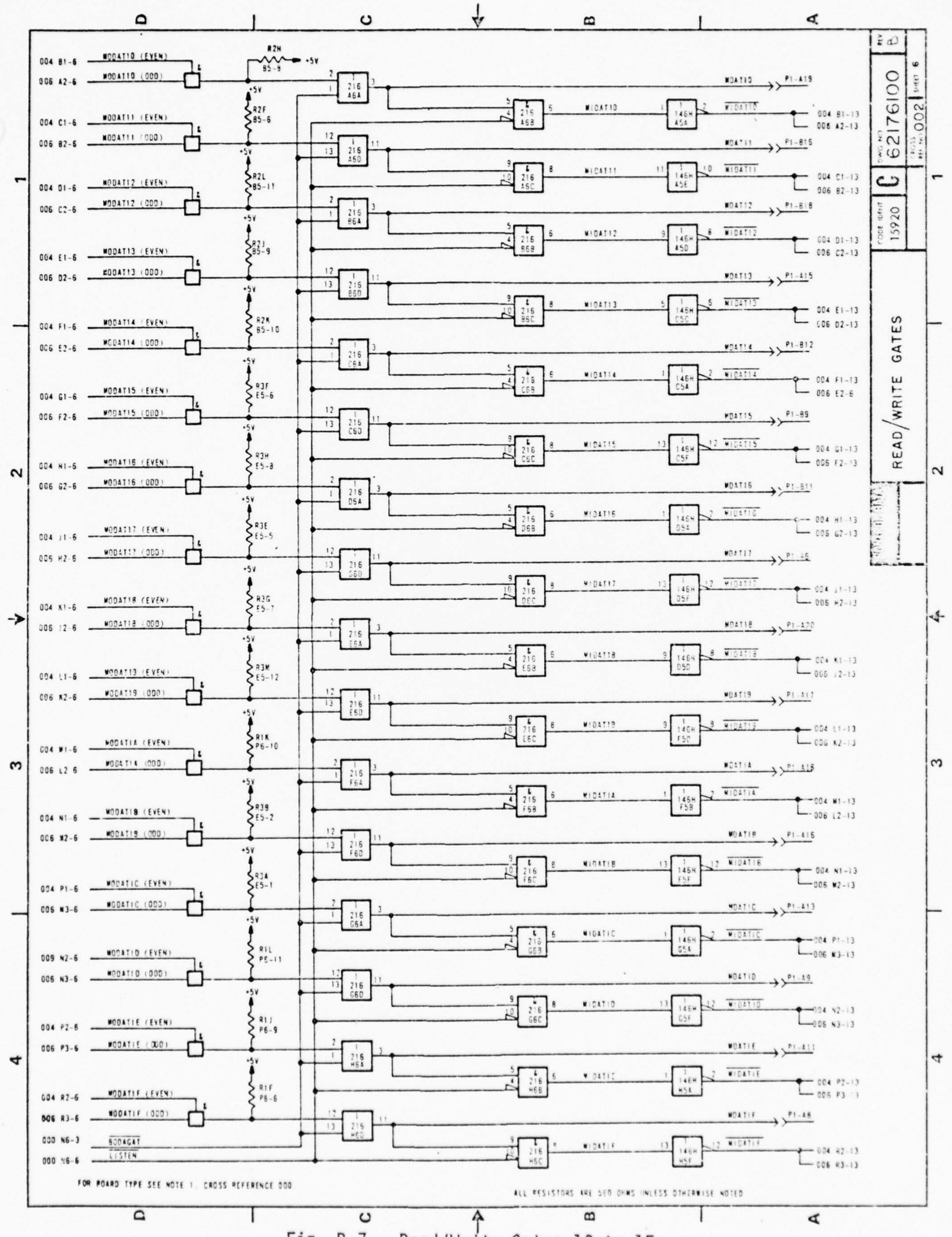

READ/WRITE GATES
62176100
15920
C
002
SHEET 6
REV B
FOR BOARD TYPE SEE NOTE 1. CROSS REFERENCE 000
ALL RESISTORS ARE 560 OHMS UNLESS OTHERWISE NOTED
BODAGAT
LISTEN

Fig. B-7. Read/Write Gates 10 to 1F

DIAGRAM 022

MEMORY ADDRESS SWITCH and MEMORY ADDRESS DRIVERS — The Memory Address Switch and the Memory Address Drivers supply a 14-bit address code, $\overline{\text{MAB00}}$ to $\overline{\text{MAB0D}}$, for selection of a 32-bit storage location in memory. The addresses come from 1) the Program Counter, 2) the Instruction Register Bits $IR10_{16}$ to IR1D, or 3) the Interrupt Encoder.

Pin 1 of the Memory Address Switch has a logical 1 to select the Program Counter Address input following a master clear or during an instruction-address *read-memory* cycle when there is no active interrupt. Pin 1 has a logical 0 to select the Instruction Register address input during the operand-derivation cycle or an instruction-execution cycle. If there is an active interrupt, the SETINT (Set Interrupt Active ff) signal is a logical 1 thus removing the enable on pin 15 of the Memory Address Switches. The memory address must now come from the Interrupt Encoder. The $\overline{\text{SETINT}}$ signal gates out the address from the Interrupt Encoder at gates G8B-6, G8A-3, and G8C-8 and forces a logical 0 at gate G8D-11. By forcing the logical 1, this is how the Interrupt Encoder lets the ACU address memory byte location 20_{16} to $3C_{16}$ for an instruction. This causes the ACU to execute one instruction that is out of sequence with the normal program.

MEMORY ADDRESSING

ACU MODE	MEMORY ADDRESS SOURCE
Master Clear Instruction Fetch (Instruction Address and Read Memory) with no Active Interrupt	Program Counter
Operand Derivation Instruction Execution	Instruction Register
Instruction Fetch (Instruction Address and Read Memory) with Active Interrupt	Interrupt Encoder

MEMORY CONTROL — The four signals, $\overline{\text{WRTBYT0}}$ to $\overline{\text{WRTBYT3}}$, let the ACU write an 8-bit data byte into memory. AOS, GP-, INP, RAD, RSB, SMP, SOC, SOS, STR, and TLM — all these instructions use Micro-Operator Bit 15 (MOB15) at gate U5B-8 to enable the output gates. The gate used depends on the contents of the C field, Bits IR1E and IR1F. The relationship between the C field and byte selection is as follows:

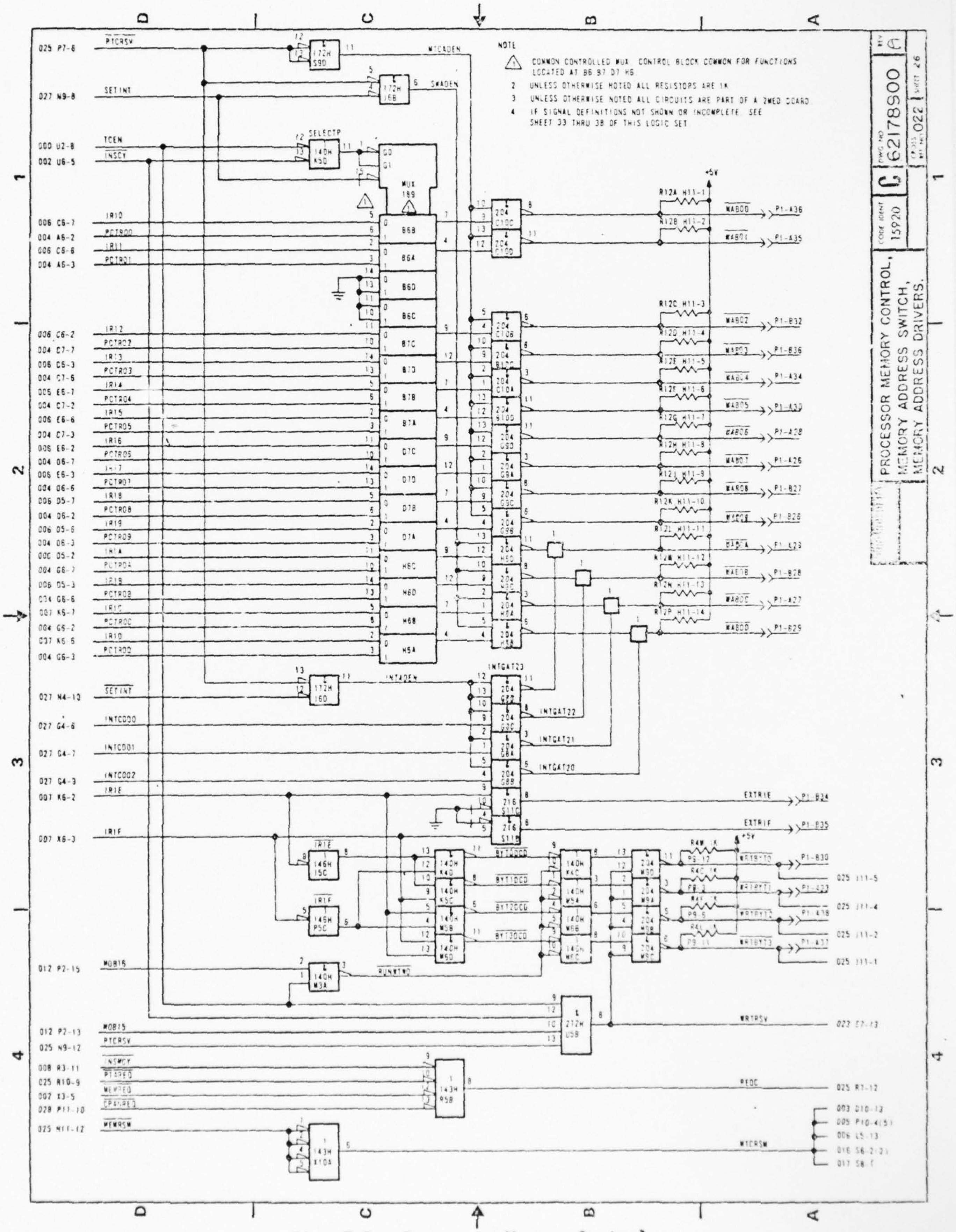
0 7	8 15	16 23	24 31
BYTE 0 C = 00	BYTE 1 C = 01	BYTE 2 C = 10	BYTE 3 C = 11

The TLM instruction uses Micro-Operator Bit 16 (MOB16) at gate M3A-3 and MOB15 at gate U5B-8 to output all Write Byte signals for a simultaneous 32-bit write to memory.

WRTRSV SIGNAL (U5B-8) — When an instruction raises the Memory Reserved for Processor and Write Decode (WRTRSV) signal by using Micro-Operator Bit 15, the logical 1 on this signal line 1) gates out the Write Byte signals $\overline{\text{WRTBYT0}}$ to $\overline{\text{WRTBYT3}}$ for data byte selection, and 2) enables the selected input at the Memory Data Switches/Drivers to appear on the Memory Data Lines for entry in memory. See also paragraph headed "Memory Control."

REQC (R5B-8) — The Memory Request Active on Port C (REQC) signal permits setting the Port C Reserved ff to gain access to either the Port C or Port A memories. The signal appears as required for any of the following 1) instruction-fetch (instruction-address and read-memory) cycle, 2) operand-derivation cycle, 3) instruction-execution cycle, and 4) control-panel request.

MTCRSM (X10A-6) — When requested data is available on the Memory Data lines, the trailing edge of the Memory Resume Buffered for Processor (MTCRSM) signal 1) loads the Auxiliary Operand Register, 2) loads the Operand Register, 3) selects the Memory Data lines at the Instruction Register switch, and 4) loads the Instruction Register with the data on the Memory Data lines.



NOTE

1. COMMON CONTROLLED MUX CONTROL BLOCK COMMON FOR FUNCTIONS LOCATED AT B6 B7 D7 H6.
2. UNLESS OTHERWISE NOTED ALL RESISTORS ARE 1K
3. UNLESS OTHERWISE NOTED ALL CIRCUITS ARE PART OF A 2MED BOARD
4. IF SIGNAL DEFINITIONS NOT SHOWN OR INCOMPLETE, SEE SHEET 33 THRU 38 OF THIS LOGIC SET.

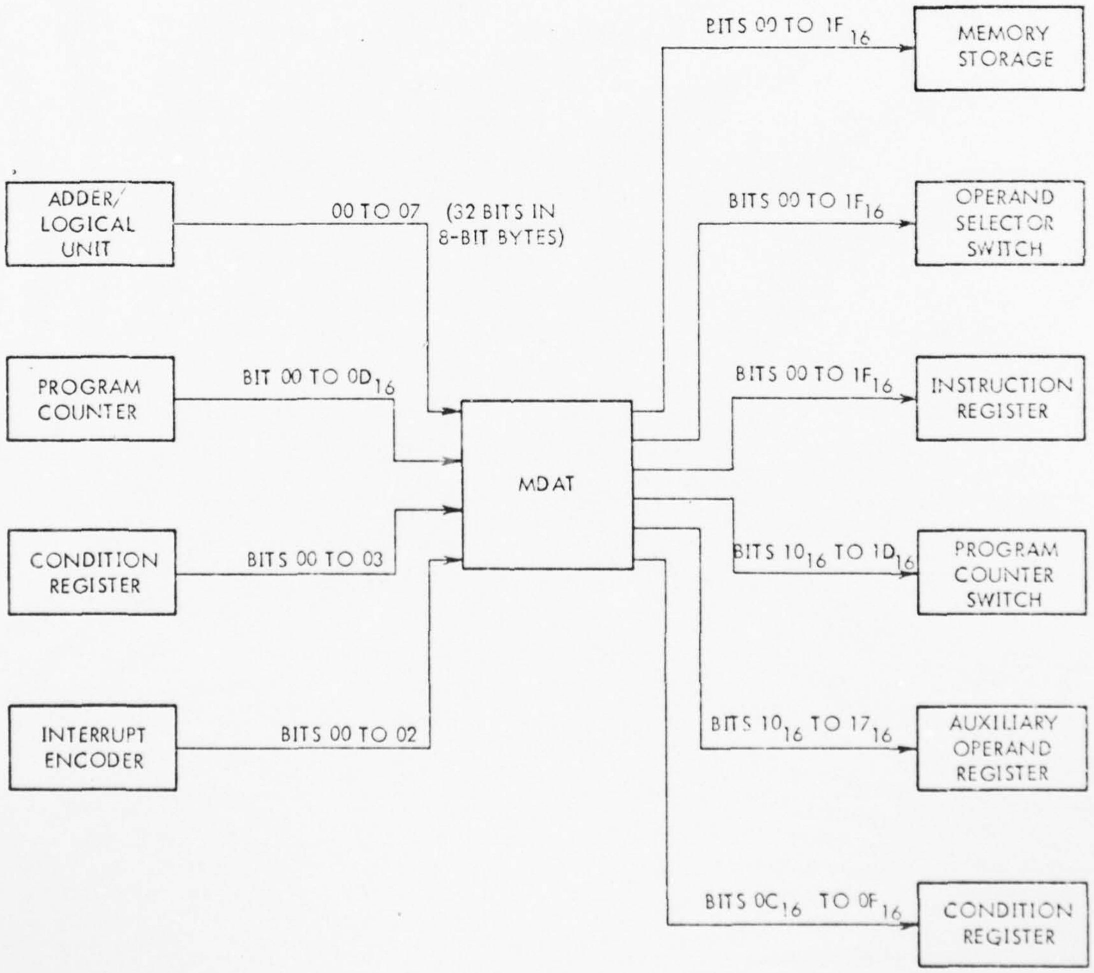
Fig. B-8. Processor Memory Control

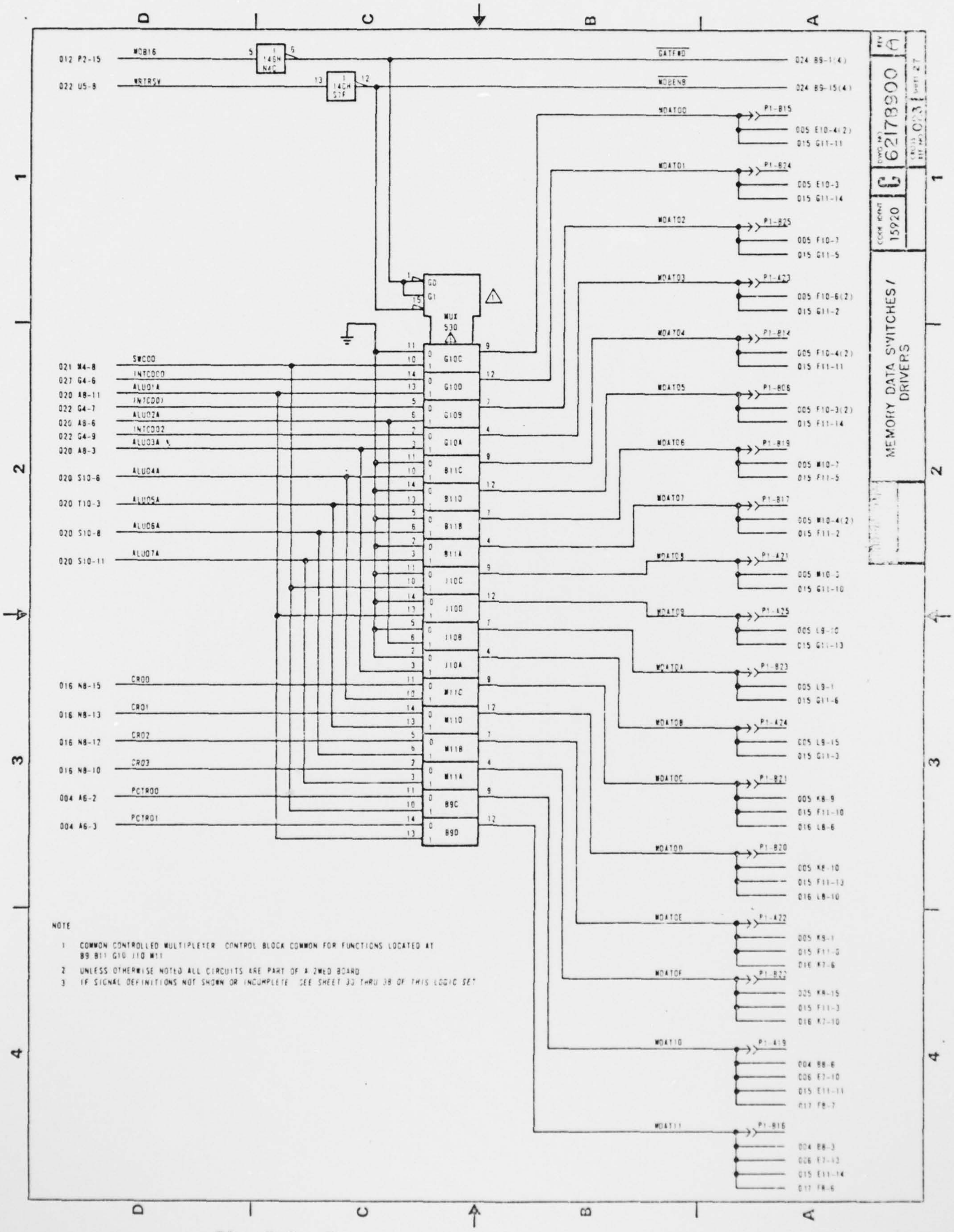
DIAGRAM 023

MEMORY DATA SWITCHES AND DRIVERS — During a memory write cycle, the Memory Address Switches, tri-state 2-input multiplexers, select for output the contents of 1) the Adder/Logical Unit, or 2) the Program Counter, the Condition Register, and the Interrupt Encoder. Switch output is to 1) the memory, 2) the Operand Selector Switch, 3) the Instruction Register, 4) the Program Counter Switch, 5) the Auxiliary Operand Register, and 6) the Condition Register.

Note that the parity control bit, SWC00, from the Parity Encoder goes to the most significant bit of each 8-bit byte of the 32-bit output. The GP- instruction uses this bit for producing odd or even parity (see paragraph headed "Parity Encoder," Diagram 021). For all other instructions, this bit is the unchanged ALU00 output of the Adder/Logical Unit.

Only the TLM instruction uses MOB16 to select the Program Counter, Condition Register, and Interrupt Encoder inputs. All other instructions select the Adder/Logical Unit inputs. Note that for the TLM, Bits 0, 4, 5, 6, 7, 8, 9, 10, 11, 30, and 31 are always logical 0's because of the grounded input.





NOTE

1 COMMON CONTROLLED MULTIPLEXER CONTROL BLOCK COMMON FOR FUNCTIONS LOCATED AT B9 B11 G10 J10 M11

2 UNLESS OTHERWISE NOTED ALL CIRCUITS ARE PART OF A 2MED BOARD

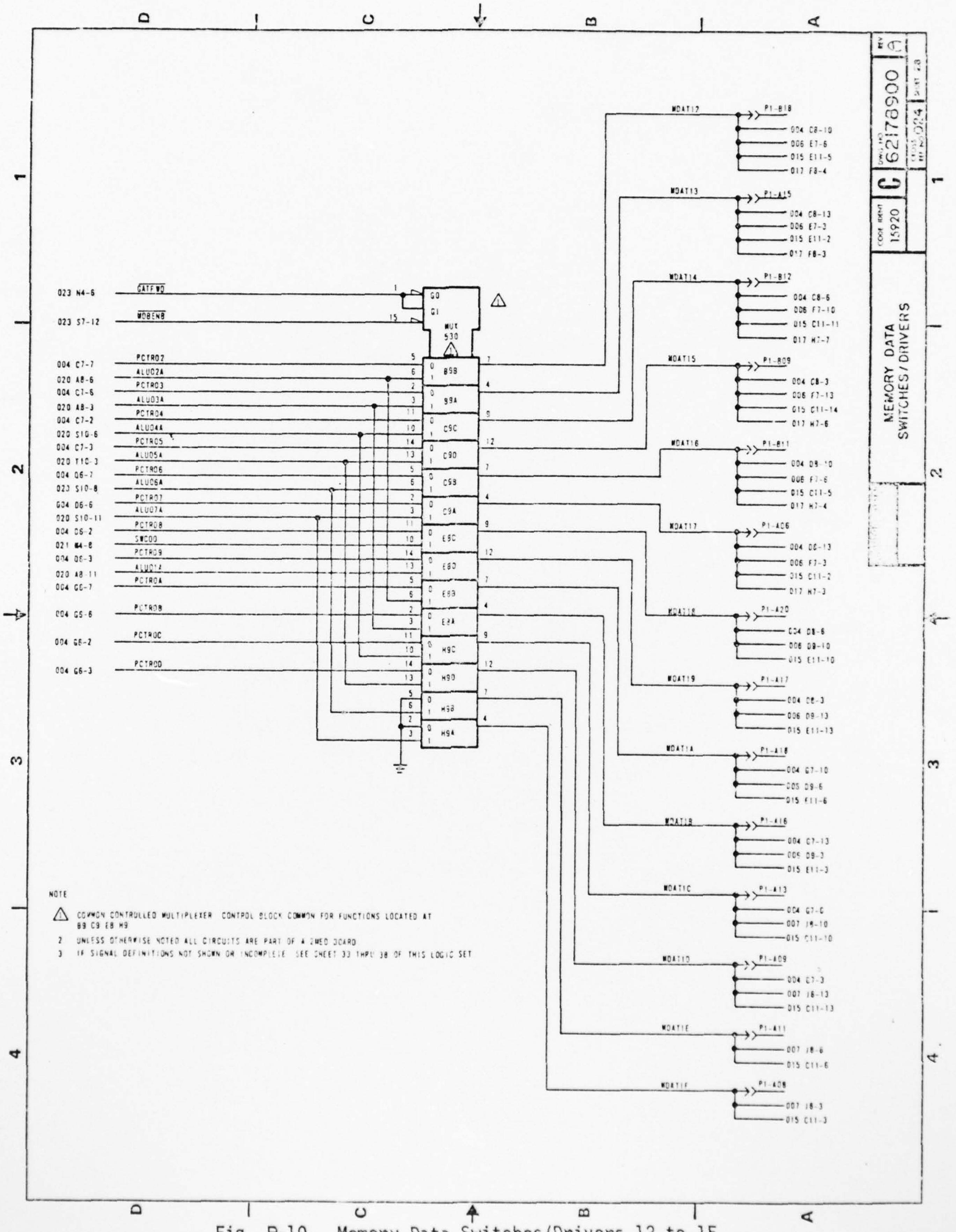
3 IF SIGNAL DEFINITIONS NOT SHOWN OR INCOMPLETE SEE SHEET 33 THRU 38 OF THIS LOGIC SET

62996100 B

Fig. B-9. Memory Data Switches/Driver 00 to 11

DIAGRAM 024

See backup page for Diagram 023.



NOTE

1. COMMON CONTROLLED MULTIPLEXER CONTROL BLOCK COMMON FOR FUNCTIONS LOCATED AT B9 C9 E8 H9
2. UNLESS OTHERWISE NOTED ALL CIRCUITS ARE PART OF A 2MED BOARD
3. IF SIGNAL DEFINITIONS NOT SHOWN OR INCOMPLETE SEE SHEET 33 THRU 38 OF THIS LOGIC SET

Fig. B-10. Memory Data Switches/Drivers 12 to 1F

$\overline{\text{REQPTA}}$ SIGNAL (V10B-3) — The I/O channel puts a logical 0 on the Request Memory Port A ($\overline{\text{REQPTA}}$) line to request access to Memory Port A (if available). This signal remains a logical 0 until the ACU responds with a logical 0 on the Memory Port A Reserved ($\overline{\text{PTARSV}}$) line to signal that Memory Port A is available. If a request for Memory Port A and a request for Memory Port C arrive simultaneously, Memory Port C has priority for the first occurrence. For all subsequent simultaneous requests, the circuit alternates access between Memory Port A and C.

$\overline{\text{PTARSV}}$ SIGNAL (J11B-8) — See paragraph headed "$\overline{\text{REQPTA}}$ Signal."

MEMORY ACCESS CONTROL (R7B, R10B, R7A, Y3A) — The Memory Access Control recognizes a memory access request, permits memory access if not busy, and responds with a status signal to show that memory access is available. Either of two signals shown here can request memory access: the first, Memory Request Active on Port C (REQC) seeks access to Memory Port C as a result of 1) an instruction-fetch read-memory cycle, 2) an operand-derivation cycle, 3) an instruction-execution cycle, and 4) a control-panel request. The second, Memory Request Active or Port A ($\overline{\text{REQPTA}}$), arrives when a Direct Memory Access Channel requests access to Memory Port A.

CIRCUIT ACTION: SIMULTANEOUS REQUESTS FOR MEMORY PORT A AND C — For the first simultaneous requests for Memory Port A and Memory Port C, the REQPTA and REQC signals appear at the set enables of the Memory Port A Request (PTAREQ) and Memory Port C Request (PTCREQ) ff's, R10B and R7B, respectively. The next negative-going PHA202 clock then sets both of these ff's. The resulting low output of pin 9 of the PTCREQ ff blocks the Start Memory Port A Request (STPTAREQ) gate P8A-12. Therefore, the STPTAREQ ff cannot set at this time to gate out the Memory Port A Reserved ($\overline{\text{PTARSV}}$) signal at gate J11B-8. Instead, the high output of pin 8 of the PTCREQ ff passes through gate P8B-6 to output the Memory Port C Reserved (PTCRSV) signal at gate N9A-12 and the $\overline{\text{PTCRSV}}$ signal at gate P7B-6. This is because the STPTAREQ ff is clear at this time thus enabling these two gates. As a result, the ACU processes the request for Memory Port C first. For all subsequent simultaneous requests, however, response to the requests alternate. This occurs because at the end of the current memory cycle, the Memory Reserved Hold (RSVHLD) ff sets to allow the next PHA202 clock to clear both the PTCREQ and STPTAREQ ff's. Now, because the PTAREQ ff is still set, supplying an enable to the STPTAREQ gate P8A-12, the high output of pin 9 of the cleared PTCREQ ff R7B pulses through the STPTAREQ gate to let the next PHA202 clock set the STPTAREQ ff. After setting, it blocks the generation of a PTCRSV signal at gate N9A and a $\overline{\text{PTCRSV}}$ signal at gate P7B, and gates out the $\overline{\text{PTARSV}}$ signal at gate J11B as a low to indicate that Memory Port A is reserved. Note that the $\overline{\text{PTAREQ}}$ signal leaves this diagram, goes to gate R5B-6, Diagram 022, and returns here as the REQC signal. This is necessary because both a request for Memory Port A and a request for Memory Port C must use the PTCREQ ff to generate the Composite Memory Request (COMREQ) signal, at gate S3C-8, for beginning a memory cycle. The COMREQ signal lets the next PHA202 clock set the Memory Lockout (MEMLO) ff. This starts the timing chain for a memory cycle.

MEMORY TIMING CIRCUIT (MEMLO, READ1, READ2, WRITE, AND RESUME FF's) — The Memory Timing circuit sequences the operation of a read or write memory cycle through use of the following signals:

REDGAT SIGNAL (N10B-6) — For a read from memory, the Memory Timing circuit sends the Read Gate (REDGAT) signal to gate the memory data from memory to the Memory Data lines.

$\overline{\text{WRTSTRO}}$ SIGNAL (N10A-3) — The Memory Timing circuit puts a logical 0 on the Write Strobe ($\overline{\text{WRTSTRO}}$) line to cause the memory to write new information at the selected location.

$\overline{\text{MEMRSM}}$ SIGNAL (N11F-12) — The Memory Resume signal goes as status to an external equipment to indicate 1) that requested memory data is on the Memory Data lines, or 2) that data on the Memory Data lines for storage is now in memory. In the ACU, it loads the Operand Register, loads the Auxiliary Operand Register, selects the Memory Data input at the Instruction Register Switch and loads the Instruction Register, and loads the addressed memory location.

RESUME SIGNAL (Y3B FF) — The Resume signal, as a logical 1, permits setting the Synchronized Memory Resume (SYCRSM) ff which, in turn, allows for the clearing of the Memory Reserved for Processor (MEMCY) ff and the Instruction-Fetch Cycle (INSCY) ff to end memory access and the instruction-fetch, when active (Diagram 002). It then allows setting the Operand-Derivation Cycle (OPSCY) ff to begin operand derivation, when required. In this circuit, it lets the Memory Reserved Hold (RSVHLD) ff set to continue the memory cycle 50 ns more before clearing the Memory Port C Reserved (PTCREQ) and Start Memory Port A Request (STPTAREQ) ff's.

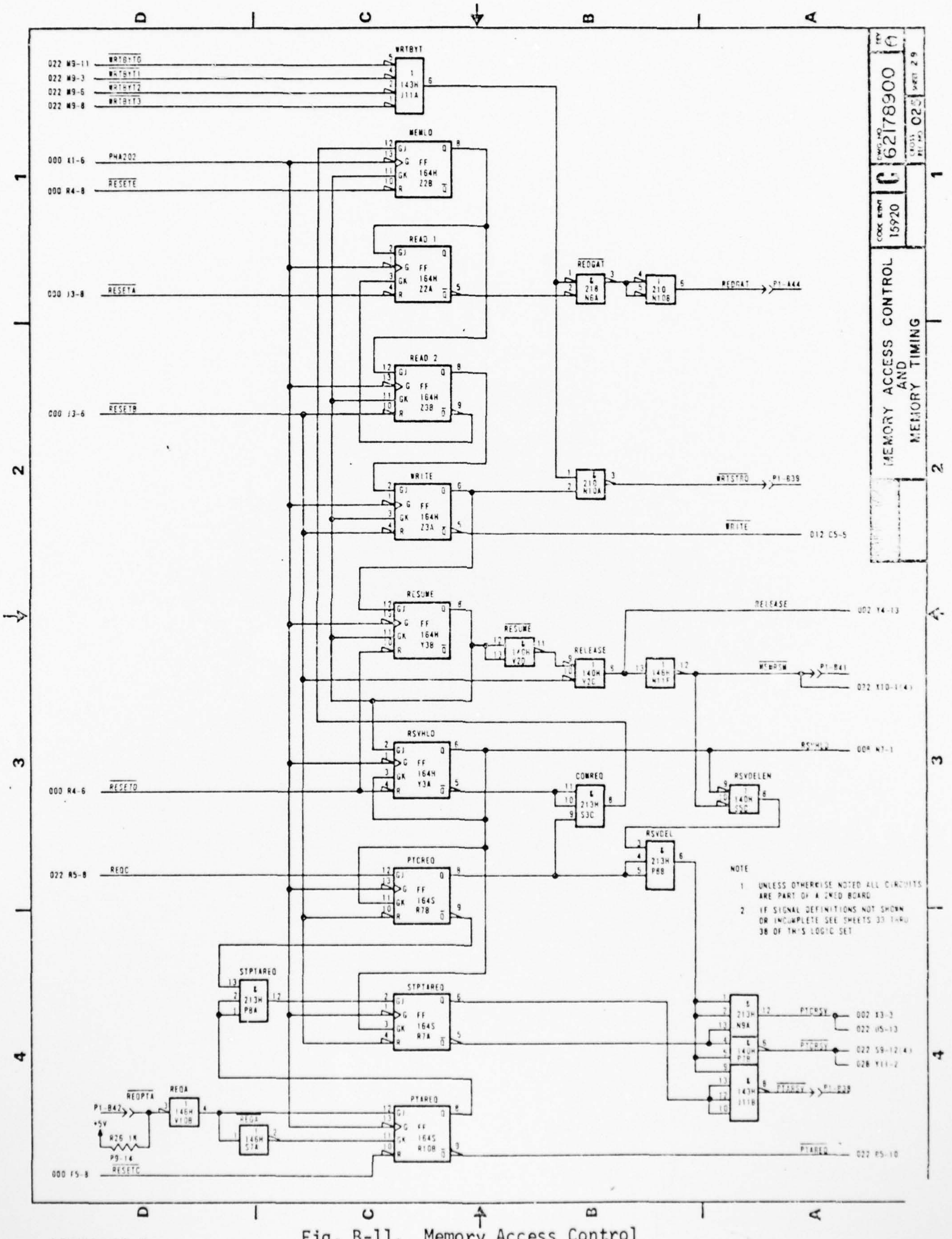


Fig. B-11. Memory Access Control

DIAGRAM 026

I/O DATA BUS INTERFACE — The I/O Data Bus has 8 two-way data lines that 1) transmit the select-channel-and-device code to each channel, 2) transmit one 8-bit byte of data to the selected channel and device, and 3) receive one 8-bit byte of information from the selected channel and device, and 4) receive a logical 0 to identify an interrupting channel. For 1) of the preceding, the channel address is on I/O Data lines 01 to 03; the device address is on lines 04 to 07.

The SEL and OUT instructions use Micro-Operator Bits 30 and 31, respectively, to control the Adder/Logical Unit gates which feed the data bus for output. The SEL instruction enables the gates for the output of the select-channel-and-device addresses. The OUT instruction does the enabling for the output of an 8-bit data byte. An INP instruction uses the I/O Data Bus to receive one 8-bit byte of information from the selected channel and device. In addition, the ACU periodically directs any interrupting channel to respond with a logical 1 on the bus line assigned to the channel. (See paragraph headed "Interrupt Encoder," Diagram 027, for channel/bus line assignment.)

Circuit output is to switch B at the Auxiliary Adder/Logical Unit input and to the Interrupt Encoder.

I/O TIMER CLOCK DIVIDERS (R1A and R2A) — If the instruction-execution cycle is doing an I/O operation, the I/O Timer Clock Dividers control the duration of the Strobe Out (STROBO) signal and the length of time the ACU waits before aborting the operation. Depending on the instruction-execution time, the Strobe Out signal lasts from 900 ns to 8.2 μs. The ACU wait or timeout is 8.2 μs (8.0 μs from the start of the Strobe Out signal).

CIRCUIT ACTION: NO RESPONSE FROM I/O CHANNEL — An OUT or INP instruction sets the Wait for I/O Return or Time Out (WAITIO) ff which lets the PHA201 clock set the Allow I/O Time Out (ALLTO) ff. The logical 1 output of this ff permits sensing for an I/O Timer run-out before receiving a Strobe In signal from the I/O channel, Diagram 027. In this diagram, it enables the clock input gate to the second of the two cascaded counters. The logical 0 output of the ff lets the first counter begin decrementing with each CLK 100 clock.

During other than an instruction-execution cycle, the pin 11 inputs of the counters have a logical 0 on them which maintains a preset count of 82_{10}. The instruction-execution cycle puts a logical 1 on these inputs to latch the count and disable the preset inputs from having any affect on the counter during the cycle.

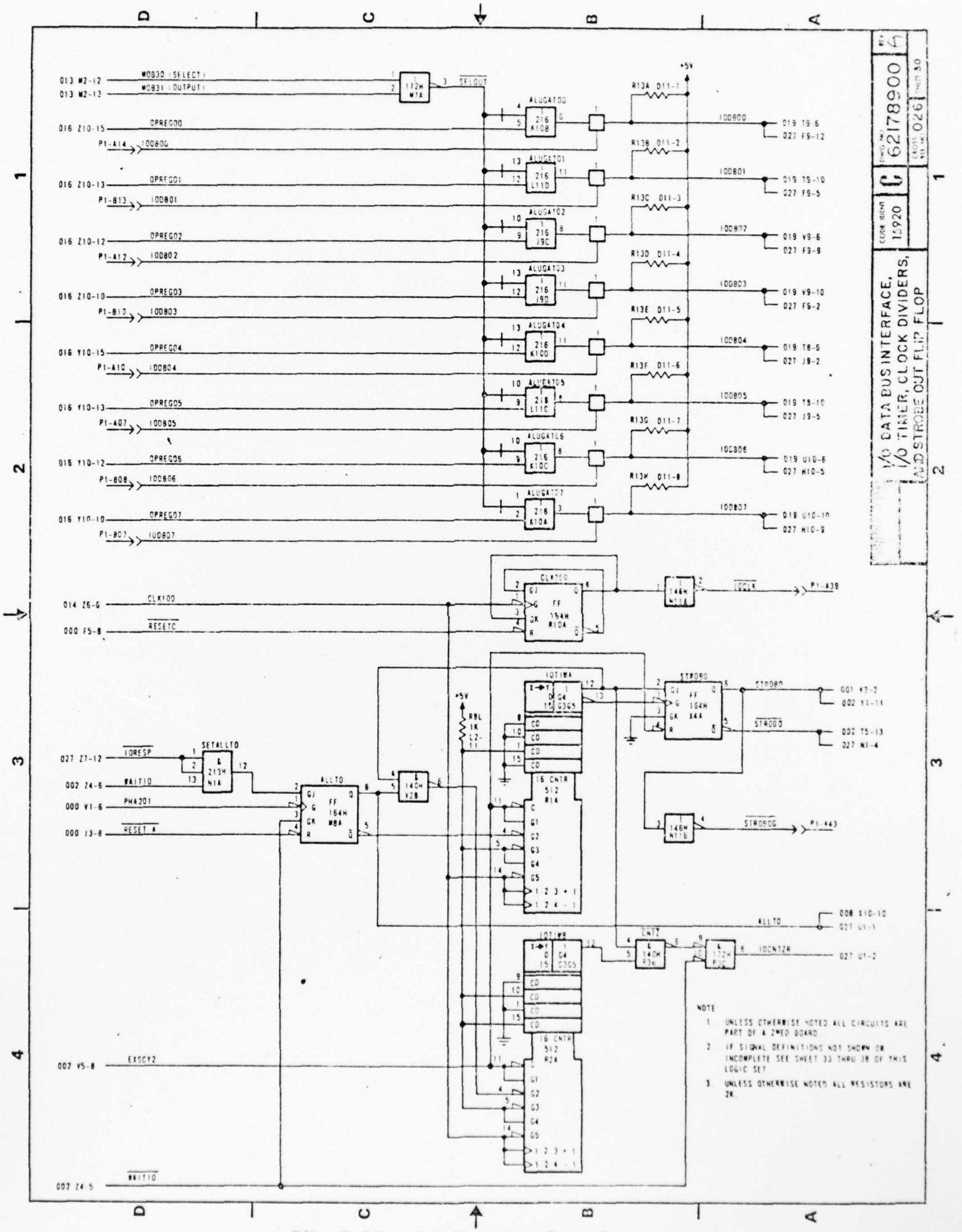
Now, the CLK 100 clock begins decrementing the counter. At a count of 80_{10}, the logical-1 pin-12 output of R1A lets the negative-going output of pin 13 set the Strobe Out (STROBO) ff thus sending the Strobe Out signal to the I/O channel. Each CLK 100 clock continues decrementing the counter.

With no response received from the I/O channel, a decrement to a count of 0 results in the I/O Count Equals Zero (IOCNTZR) signal at gate P3C-8, which lets the next PHA201 clock set the Cycle Counter Equals Zero (CYCNTZR) ff to output the I/O Reject or Time Out (IOREJTO) signal at gate W3C-8 (Diagram 027). This signal lets the next PHA202 clock set the I/O Abort ff (Diagram 002). The output of this ff, in turn, permits PHA201 to clear the Instruction-Execution Cycle (EXSCY) ff and thereby end the Strobe Out signal by sending a low here on the EXSCY2 signal line to clear the Strobe Out ff. The low on the EXSCY2 line also presets the I/O Timer to a count of 82_{10}. In addition, the low on the EXSCY2 line of Diagram 002 also clears the Wait for I/O Return or Time Out (WAITIO) ff in the same diagram. This places a high on the $\overline{\text{WAITIO}}$ line at the clear enable of the Allow Time Out (ALLTO) ff, here. As a result, the next PHA201 clears the Allow Time Out ff and ends the decrementing of the I/O Timer.

If the ACU receives a Strobe In response from a channel, the Strobe In (IOSTROBI) ff sets (Diagram 027) and outputs the I/O Response (IORESP) signal, via gate Z7A-12, to allow for clearing of the Wait for I/O Return or Time Out ff (Diagram 002). The resulting low on the $\overline{\text{WAITIO}}$ line, here, clears the Allow Time Out ff to stop the I/O Timer.

If the ACU receives an I/O Reject signal from a channel, it results in clearing the Allow Time Out ff to stop the I/O Timer with action similar to that which follows reception of a Strobe In response. Furthermore, it causes an abort of the I/O operation as if the I/O Timer had run out. The I/O Reject or Time Out (IOREJTO) signal of Diagram 027 raises after the setting of the I/O Reject (IOREJC) ff to clear the I/O Abort ff (Diagram 002). Thus, the ACU abandons the I/O operation and begins a new instruction fetch.

CLK200 FF (R10A) — The Clock 200 ff supplies a 200-nsec, 5-MHz signal to the interface.



I/O DATA BUS INTERFACE, I/O TIMER, CLOCK DIVIDERS, AND STROBE OUT FLIP FLOP
15920
C
62178900
026
ALUGAT00
ALUGAT07
CLKT00
IOTIMA
IOTIMB
STROBO
ALLTO
SETALLTO
NOTE
1. UNLESS OTHERWISE NOTED ALL CIRCUITS ARE PART OF A 2MED BOARD.
2. IF SIGNAL DEFINITIONS NOT SHOWN OR INCOMPLETE SEE SHEET 33 THRU 38 OF THIS LOGIC SET.
3. UNLESS OTHERWISE NOTED ALL RESISTORS ARE 2K.

Fig. B-12. I/O Data Bus Interface

DIAGRAM 027

I/O RESPONSE FF's: IOSTROBI, IOREJC, and CYCNTZR (Y2A, Y2B, and V3B) — During either an INP or OUT instruction, the ACU looks for a normal response from the I/O channel that indicates that the I/O channel either accepted or rejected the function. Normal response for an INP instruction is either a Strobe In signal indicating that the requested data is on the I/O Data bus, or a Reject signal if the channel is busy and cannot respond with the data. Normal response for an OUT instruction is either a Strobe In signal when the channel accepts the data from the ACU, or a Reject signal if it is busy and cannot accept the data. If the ACU receives no response, it will abort the I/O function after 8.2 μs (8 μs after the leading edge of the Strobe Out signal).

Therefore, during an input or output to an I/O channel, the ACU sets the Allow I/O Time Out ff thus permitting gate U1A to sense for a time out while waiting for either a Strobe In or a Reject signal response from the I/O channel. If there is a Strobe In response, the I/O Strobe In ff sets and outputs a logical 1 on pin 6 that advances the Program Counter by 1 for a program exit to P plus 2. The low at the pin 5 output enables gate X10B for a memory cycle (Diagram 008), and outputs the $\overline{\text{I/O Response}}$ ($\overline{\text{IORSP}}$) signal at gate Z7A to clear the Wait for I/O Return or Time Out (WAITIO) ff of Diagram 002 to stop the run out of the I/O Timer.

If the response is a Reject signal, the I/O Reject (IOREJC) ff sets and uses the logical 1 output on pin 8 to 1) cause an I/O abort, via gate W3C-8; 2) clear the Condition Register, via gate W3C-8; and 3) clear the Wait for I/O Return or Time Out ff via gate Z7A. This last action clears the Allow Time Out ff to stop the I/O Timer run out.

If the ACU receives no response from the I/O channel within 8.2 μsec, the I/O Timer runs out and sends the I/O Counter Equals Zero (IOCNTZR) signal to let the PHA201 clock signal set the Cycle Counter Equals Zero (CYCNTZR) ff. This causes an I/O abort, master clears the Condition Register, and clears the Wait for I/O Return or Time Out ff and the Allow I/O Time Out ff, via gates W3C-8 and Z7A, respectively.

INTERRUPT REGISTER (F4 AND G5) — When a channel sends an interrupt on the $\overline{\text{Channel Interrupt Request Active}}$ line, it also puts a logical 0 on the I/O Data line assigned to the channel to identify itself. The Interrupt Register, two 4-bit latches, loads these logical 0's for use by the Interrupt Encoder during an interrupt routine (see paragraph headed, "Interrupt Control"). The following table lists the channels and their I/O Data line assignments.

INTERRUPT ON CHANNEL	I/O DATA BUS BIT	BINARY-WEIGHTED OUTPUT
0 (HIGHEST PRIORITY)	IODB00	000
1	IODB01	001
2	IODB02	010
3	IODB03	011
4	IODB04	100
5	IODB05	101
6	IODB06	110
7 (LOWEST PRIORITY)	IODB07	111

INTERRUPT ENCODER (G4A) — The Interrupt Encoder, an 8-input priority encoder, outputs a binary-weighted code for the highest order input. Pin 4 has the highest priority, and Pin 10 has the lowest. The preceding table shows the relationship of the interrupt channel, the I/O Data Bus Bit, and the binary-weighted output.

Depending on the instruction in the memory-trap location, control of the program sequence follows one of these three variations:

1) For all instructions except TLM, RTR, and UTR — the ACU executes one instruction out of sequence with the normal program and, then, returns to the normal program sequence.

2) For a TLM instruction only, the ACU stores, in the Condition Register, P plus 1 for the last instruction executed when the ACU detected the Interrput Request; goes to the memory location that the TLM specifies; executes the subroutine stored there; and returns to the normal program sequence. The TLM instruction must disable the interrupt system and thus enable the Memory Address Switches for normal memory addressing during the subroutine. At the completion of the subroutine, an RTR instruction returns control to the normal program sequence and normally reenables the interrupt system.

3) For RTR, UTR, RPT, and TOC (for RPT and TOC only if transfer condition is met) — the ACU goes to the memory location that these instructions specify and begins executing a new program from this location. Control does not return to the original program sequence. If the transfer condition is not met for an RPT or TOC instruction, the ACU executes one instruction out of sequence with the normal program and, then, returns to the normal program sequence.

OPCYC SIGNAL (S10A-3) — The Operand-Derivation Cycle signal goes to the I/O channels as a status signal, which indicates that the ACU is currently deriving an operand.

INSCYC SIGNAL (K11B-4) — The Instruction-Fetch Cycle signal goes to the I/O channels as a status signal which indicates that the ACU is presently getting a new instruction.

SELCH SIGNAL

FUNT 0 to FUNT 2 SIGNAL

INOUT SIGNAL

IORESET SIGNAL

} See "Signal Definitions," Section 2, Operation and Programming

The binary-weighted output then goes to the Interrupt Gating of the Memory Address lines where it combines with a forced Bit 2^3. This lets the Interrupt Encoder address memory-trap locations 08_{16} through $0F_{16}$ for one instruction.

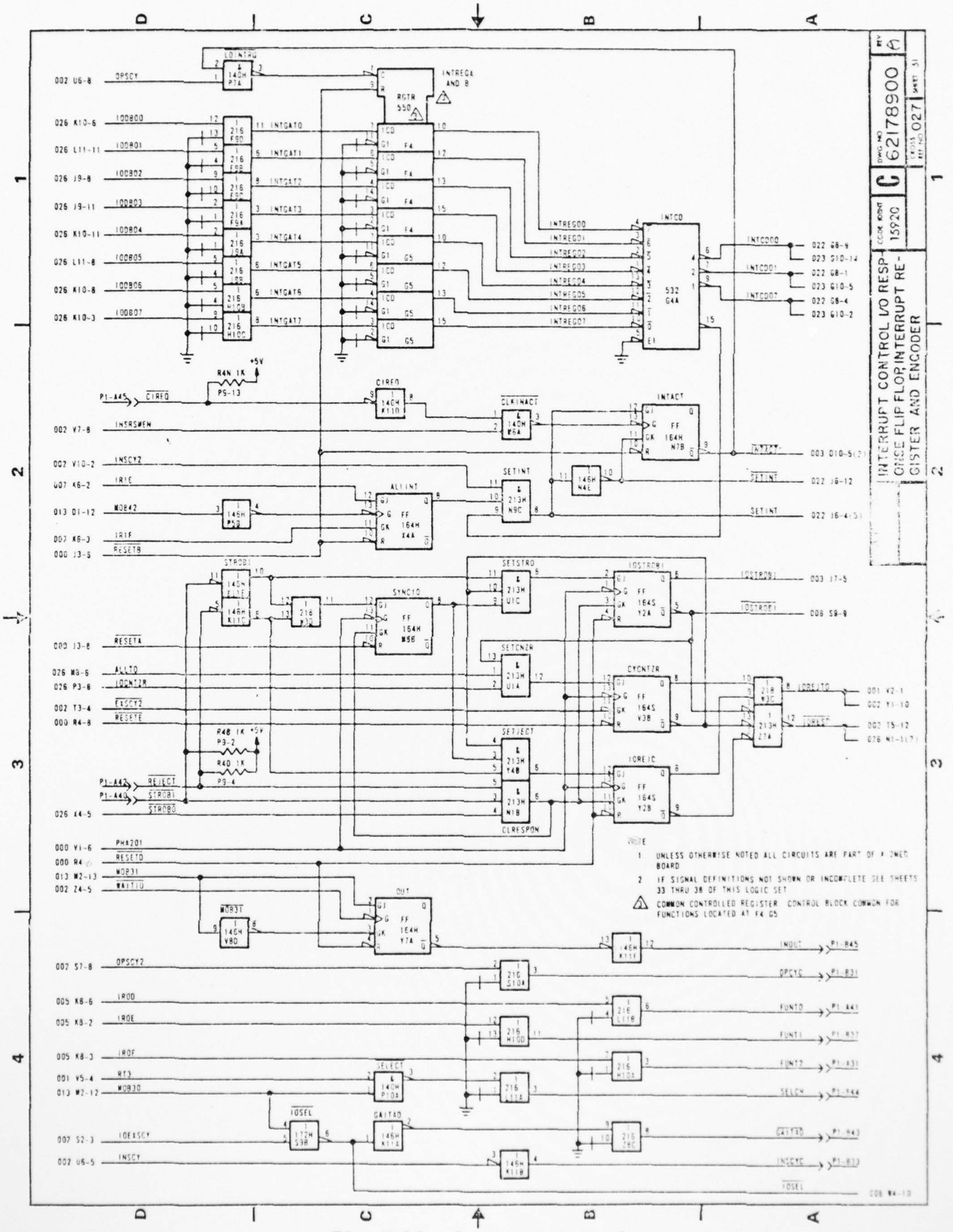
INTERRUPT CONTROL: The ACU controls two interrupt circuits. One is in a channel. The other is in the ACU and appears in this diagram. If a channel interrupt system is enabled, via an SEL instruction, and the data interrupts of the channel are enabled, via an OUT instruction with a Control-1 function — the channel can send an interrupt to the ACU when it requires service. This interrupt arrives here as a low on the $\overline{\text{Channel Interrupt Request Active}}$ ($\overline{\text{CIREQ}}$) line. Whether the ACU services the interrupt depends on wherher it enabled its own interrupt system.

Enabling the ACU interrupt system occurs as follows. First, an RTR or UTR control-transfer instruction with Bit 30, IR1E, high and Bit 31, IRIF, low enables setting the Allow Interrupt ff X4A when one of these instructions loads into the Instruction Register. Later, when the ACU leaves the operand-derivation cycle and enters the instruction-execution cycle, Micro-Operator Bit 42 (MOB 42) appears and toggle-sets the Allow Interrupt ff. (Instructions RTR and UTR raise MOB42 in the Micro-Operator Memory.) After the Allow Interrupt ff sets, the Set Interrupt Active gate N9C-8 waits for an interrupting channel to send its address on one of the I/O Data lines. The resulting low on one of these lines produces a high on output pin 15 of the Interrupt Encoder G4A. Now, the gate waits for the beginning of the next instruction-address and read-memory cycle. This sends the INSCY2 signal to the gate, placing a set enable on the Interrupt Active ff. The output of the gate also disables input selection by the Memory Address Switches (Diagram 022) and senses for the Memory Port C Reserved (PCTRSV) signal to know that Memory Port C is available before it gates out the memory-trap addressing from the interrupt gating to the Memory Address bus.

Now, with the $\overline{\text{Channel Interrupt Request Active}}$ line low, the $\overline{\text{Clock Interrupt Active}}$ gate M6A-3 waits for the completion of the current instruction memory cycle. At this time, the Instruction Resume Enable (INSRSMEN) signal passes through the gate and sets the Interrupt Active ff. The low output of pin 9 of the ff blocks advance of the Program Counter to the next effective address. The output of this gate also blocks the Load Interrupt Register gate P7A-3 so that if another higher priority channel interrupt occurs at this time, it does not reach the Interrupt Encoder and thereby change its output coding. The Instruction Register loads the instruction at the memory-trap location. If the instruction is a TLM, it must disable the interrupt system. Doing this allows the Memory Data switches to output the contents of the Program Counter to select the instructions of the following subroutine. The TLM instruction specifies the location in memory of the first instruction of the subroutine.

As described in the preceding, when the ACU services an active interrupt, it executes the instruction stored at the memory-trap location assigned to the interrupting channel. The following shows the relationship between the interrupting channel and its assigned memory-trap location.

INTERRUPTING CHANNEL	MEMORY-TRAP ADDRESS (Memory Address Bus) 0A	0B	0C	0D
0	1	0	0	0
1	1	0	0	1
2	1	0	1	0
3	1	0	1	1
4	1	1	0	0
5	1	1	0	1
6	1	1	1	0
7	1	1	1	1



NOTE

1 UNLESS OTHERWISE NOTED ALL CIRCUITS ARE PART OF A 2WED BOARD

2 IF SIGNAL DEFINITIONS NOT SHOWN OR INCOMPLETE SEE SHEETS 33 THRU 38 OF THIS LOGIC SET

△ COMMON CONTROLLED REGISTER CONTROL BLOCK COMMON FOR FUNCTIONS LOCATED AT F4 G5

Fig. B-13. Interrupt Control

Appendix C

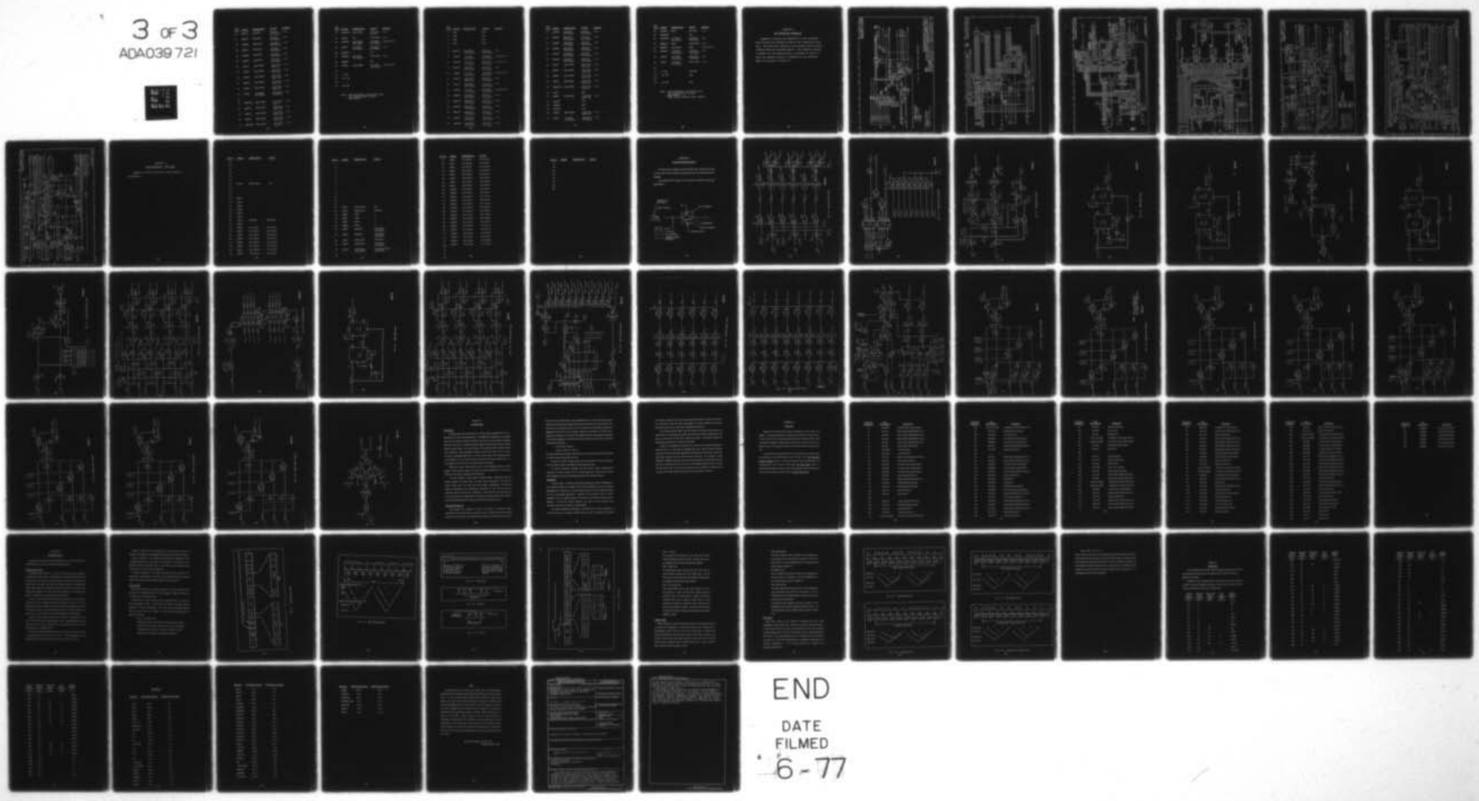
ACU Connector P1 Pin Assignments

PIN P1-A	SIGNAL	TERMINATION	ORIGIN	LOADING
1	+5 VDC		MTC	
2	+5 VDC		MTC	
3	+5 VDC		MTC	
4	+5 VDC		MTC	
5				
6	MDAT17	Mem/8093	ACU/Mux Tri-State	1 UL
7	IODB05	I/O Mux Tri-State	ACU/8093	2 UL-2K P/U
8	MDAT1F	Mem/8093	ACU/Mux Tri-State	1 UL
9	MDAT1D	Mem/8093	ACU/Mux Tri-State	1 UL
10	IODB04	I/O Mux Tri-State	ACU/8093	2 UL-2K P/U
11	MDAT1E	Mem/8093	ACU/Mux Tri-State	1 UL
12	IODB02	I/O Mux Tri-State	ACU/8093	2 UL-2K P/U
13	MDAT1C	Mem/8093	ACU/Mux Tri-State	1 UL
14	IODB00	I/O Mux Tri-State	ACU/8093	2 UL-2K P/U
15	MDAT13	Mem/8093	ACU/Mux Tri-State	1 UL
16	MDAT1B	Mem/8093	ACU/Mux Tri-State	1 UL
17	MDAT19	Mem/8093	ACU/Mux Tri-State	1 UL
18	MDAT1A	Mem/8093	ACU/Mux Tri-State	1 UL
19	MDAT10	Mem/8093	ACU/Mux Tri-State	1 UL

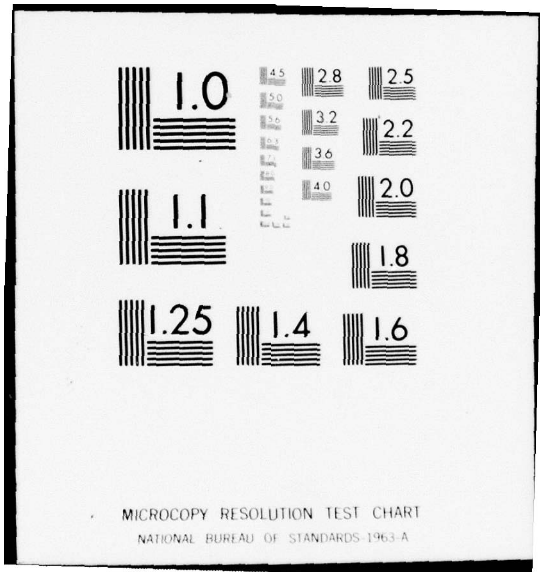
AD-A039 721   AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO   SCH--ETC   F/G 9/2
DESIGN OF A FLOPPY DISK DRIVE CONTROLLER INTERFACE FOR THE 6507--ETC(U)
MAR 77   M J VARNER

UNCLASSIFIED   AFIT-GCS/EE/77-2   NL

3 OF 3
ADA039721



END
DATE
FILMED
6-77


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

PIN				
P1-A	SIGNAL	TERMINATION	ORIGIN	LOADING
20	MDAT18	Mem/8093	ACU/Mux Tri-State	1 UL
21	MDAT08	Mem/8093	ACU/Mux Tri-State	1 UL
22	MDAT0E	Mem/8093	ACU/Mux Tri-State	1 UL
23	MDAT03	Mem/8093	ACU/Mux Tri-State	1 UL
24	MDAT0B	Mem/8093	ACU/Mux Tri-State	1 UL
25	MDAT09	Mem/8093	ACU/Mux Tri-State	1 UL
26	MAB07/	Mem/74H04	ACU/7438 with P/U	1 UL
27	MAB0C/	Mem/74H04	ACU/7438 with P/U	1 UL
28	MAB06/	Mem/74H04	ACU/7438 with P/U	1 UL
29	MAB0A/	Mem/74H04	ACU/7438 with P/U	1 UL
30	MAB05/	Mem/74H04	ACU/7438 with P/U	1 UL
31	FUNT2	I/O-8093 Tri-State	ACU/8093 Tri-State	1 UL
32				
33	WRTBYT1/	Mem/74H04	ACU/7438 1K P/U	1 UL
34	MAB04/	Mem/74H04	ACU/7438 with P/U	1 UL
35	MAB01/	Mem/74H04	ACU/7438 with P/U	1 UL
36	MAB00/	Mem/74H04	ACU/7438 with P/U	1 UL
37	WRTBYT3/	Mem/74H04	ACU/7438 1K P/U	1 UL

PIN P1-A	SIGNAL	TERMINATION	ORIGIN	LOADING
38	WRTBYT2/	Mem/74H04	ACU/7438 1K P/U	1 UL
39	IOCLK/	TTL	ACU/74H04	1 UL
40	STROBI/	ACU/74H11 ACU/74H04	I/O-8093 Tri-State	2 UL-1K P/U
41	FUNTO	I/O-8093 Tri-State	ACU/8093 Tri-State	1 UL
42	REJECT/		I/O	
43	STROBO/	I/O-8093 Tri-State	ACU/74H04	1 UL
44	REDGAT		ACU	
45	CIREQ/	ACU/74H04	I/O-8093 Tri-State	1 UL-1K P/U
46				
47	-5 VDC			
48	-16 VDC			
49				
50	+16 VDC			

NOTE: ACU--Arithmetic and Control Unit
I/O--Input/Output Channel
Mem--Memory

PIN P1-B	SIGNAL	TERMINATION	ORIGIN	LOADING
1	GRD		MTC	
2	GRD		MTC	
3	GRD		MTC	
4	GRD		MTC	
5				
6	MDAT05	Mem/8093 Tri-State	ACU/Mux Tri-State	1 UL
7	IODB07	I/O Mux Tri-State	ACU/8093 Tri-State	2 UL-2K P/U
8	IODB06	I/O Mux Tri-State	ACU/8093 Tri-State	2 UL-2K P/U
9	MDAT15	Mem/8093 Tri-State	ACU/Mux Tri-State	1 UL
10	IODB03	I/O Mux Tri-State	ACU/8093 Tri-State	2 UL-2K P/U
11	MDAT16	Mem/8093 Tri-State	ACU/Mux Tri-State	1 UL
12	MDAT14	Mem/8093 Tri-State	ACU/Mux Tri-State	1 UL
13	IODB01	I/O Mux Tri-State	ACU/8093 Tri-State	2 UL-2K P/U
14	MDAT04	Mem/8093 Tri-State	ACU/Mux Tri-State	1 UL
15	MDAT00	Mem/8093 Tri-State	ACU/Mux Tri-State	1 UL
16	MDAT11	Mem/8093 Tri-State	ACU/Mux Tri-State	1 UL
17	MDAT07	Mem/8093 Tri-State	ACU/Mux Tri-State	1 UL
18	MDAT12	Mem/8093 Tri-State	ACU/Mux Tri-State	1 UL
19	MDAT06	Mem/8093 Tri-State	ACU/Mux Tri-State	1 UL

PIN P1-B	SIGNAL	TERMINATION	ORIGIN	LOADING
20	MDAT0D	Mem/8093 Tri-State	ACU/Mux Tri-State	1 UL
21	MDAT0C	Mem/8093 Tri-State	ACU/Mux Tri-State	1 UL
22	MDAT0F	Mem/8093 Tri-State	ACU/Mux Tri-State	1 UL
23	MDAT0A	Mem/8093 Tri-State	ACU/Mux Tri-State	1 UL
24	MDAT01	Mem/8093 Tri-State	ACU/Mux Tri-State	1 UL
25	MDAT02	Mem/8093 Tri-State	ACU/Mux Tri-State	1 UL
26	MAB09	Mem/74H04	ACU/7438 with P/U	1 UL
27	MAB08	Mem/74H04	ACU/7438 with P/U	1 UL
28	MAB0B	Mem/74H04	ACU/7438 with P/U	1 UL
29	MAB0D	Mem/74H04	ACU/7438 with P/U	1 UL
30	WRTBYT0/	Mem/74H04	ACU/7438 1K P/U	1 UL
31	OPCYC		ACU	
32	MAB02	Mem/74H04	ACU/7438 with P/U	1 UL
33	INSCYC		ACU	
34	EXTR1E		ACU	
35	EXTR1F		ACU	
36	MAB03	Mem/74H04	ACU/7438 with P/U	1 UL
37	FUNT1	I/O-8093 Tri-State	ACU/8093 Tri-State	1 UL

PIN P1-B	SIGNAL	TERMINATION	ORIGIN	LOADING
38	PTARSV/	DMAC	ACU/TTL	1 UL
39	WRTSTRO/		ACU	
40	IORESET/	I/O-8093 Tri-State	ACU/8093 Tri-State	1 UL
41	MEMRSM/	DMAC	ACU/74H04	1 UL
42	REQPTA	ACU/74H04	DMAC	1 UL-1K P/U
43	GAITAD/	I/O-8093 Tri-State	ACU-8093 Tri-State	1 UL
44	SELCH/	I/O-8093 Tri-State	ACU/8093 Tri-State	1 UL
45	INOUT	I/O-8093 Tri-State	ACU/74H04	1 UL
46				
47	-5 VDC		ACU/MTC	
48	-16 VDC		MTC	
49				
50	+16 VDC		MTC	

NOTE: ACU--Arithmetic and Control Unit
I/O--Input/Output Channel
Mem--Memory
DMAC--Direct Memory Access Channel

Appendix D

DOS Controller Schematics

Appendix D contains the schematics of the controller which controls the storage of data on the floppy disk drive unit. The controller consists of two printed circuit boards (Channel Board and Interface Board). The schematic drawing (4 sheets) for the Channel Board is provided in Figure D-1 while the schematic drawing (3 sheets) for the Interface Board is provided in Figure D-2.

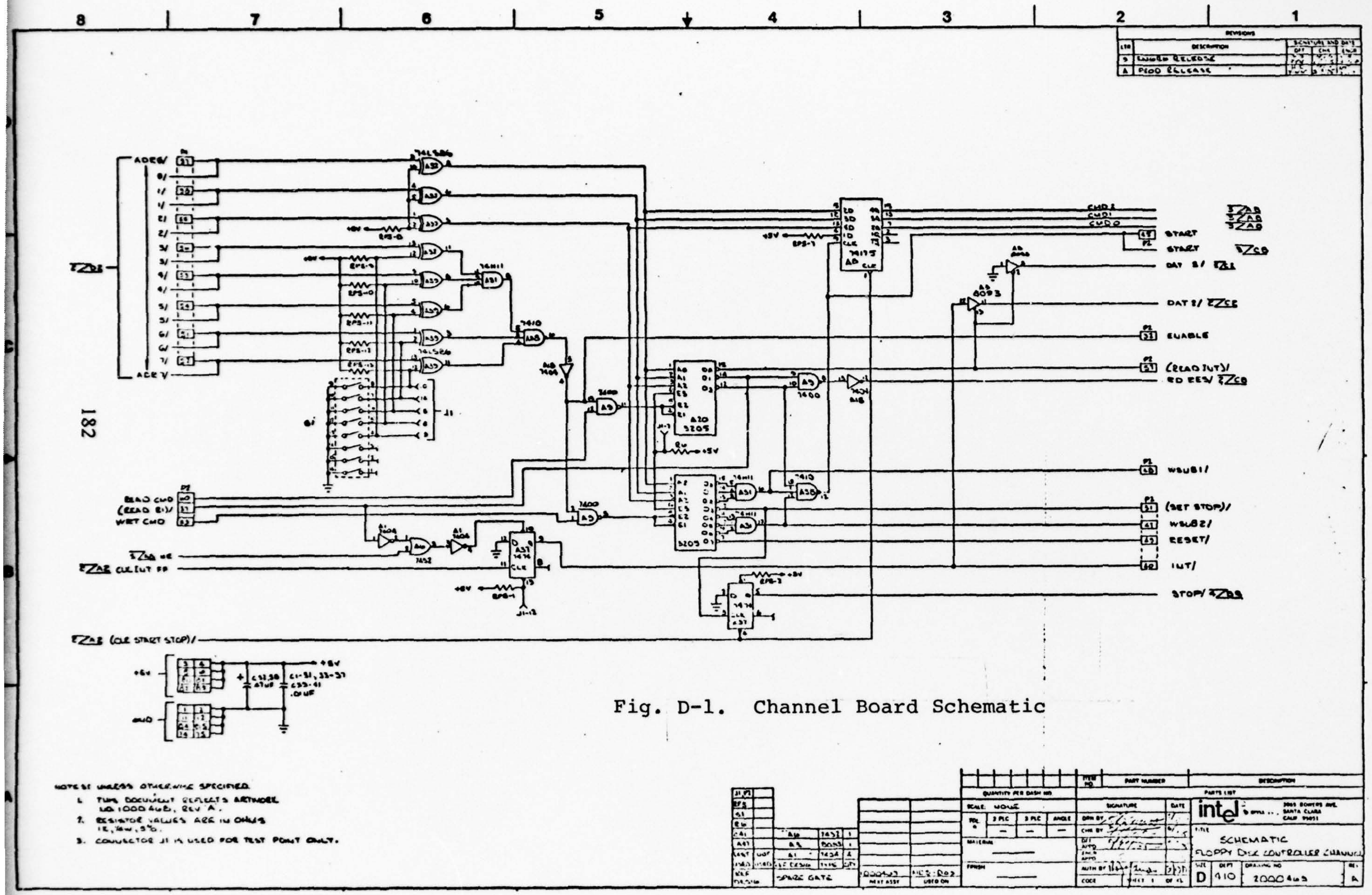
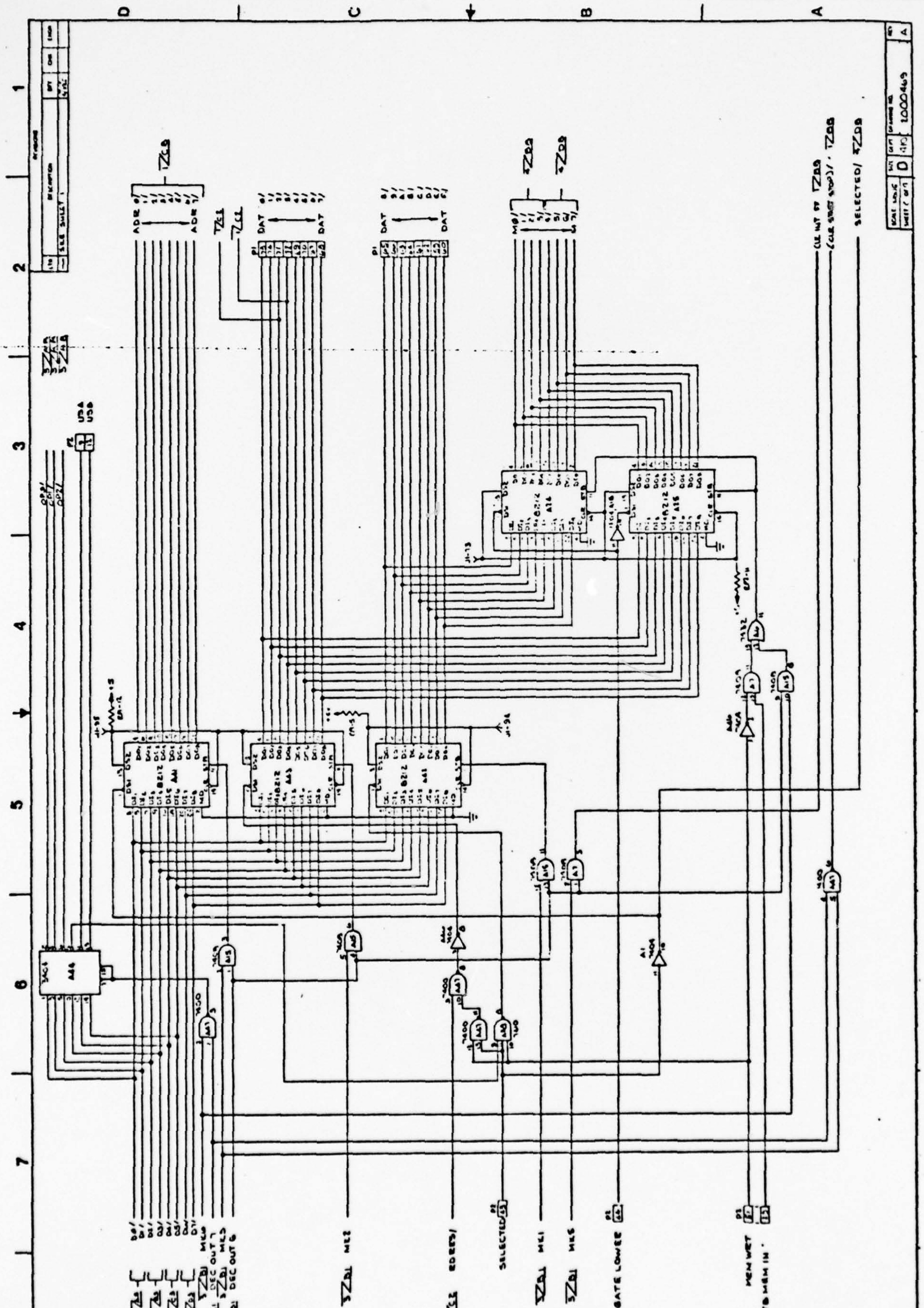
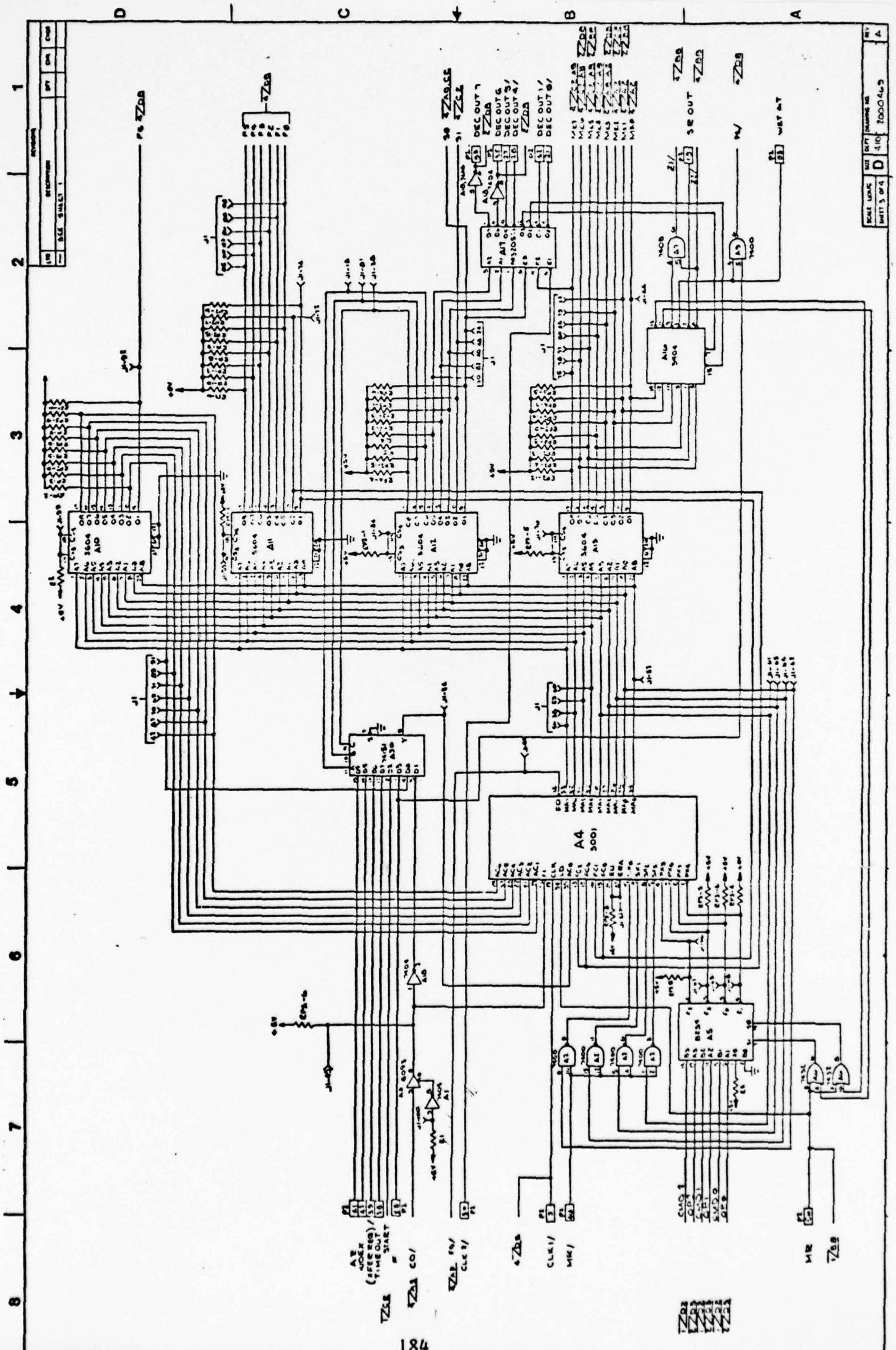
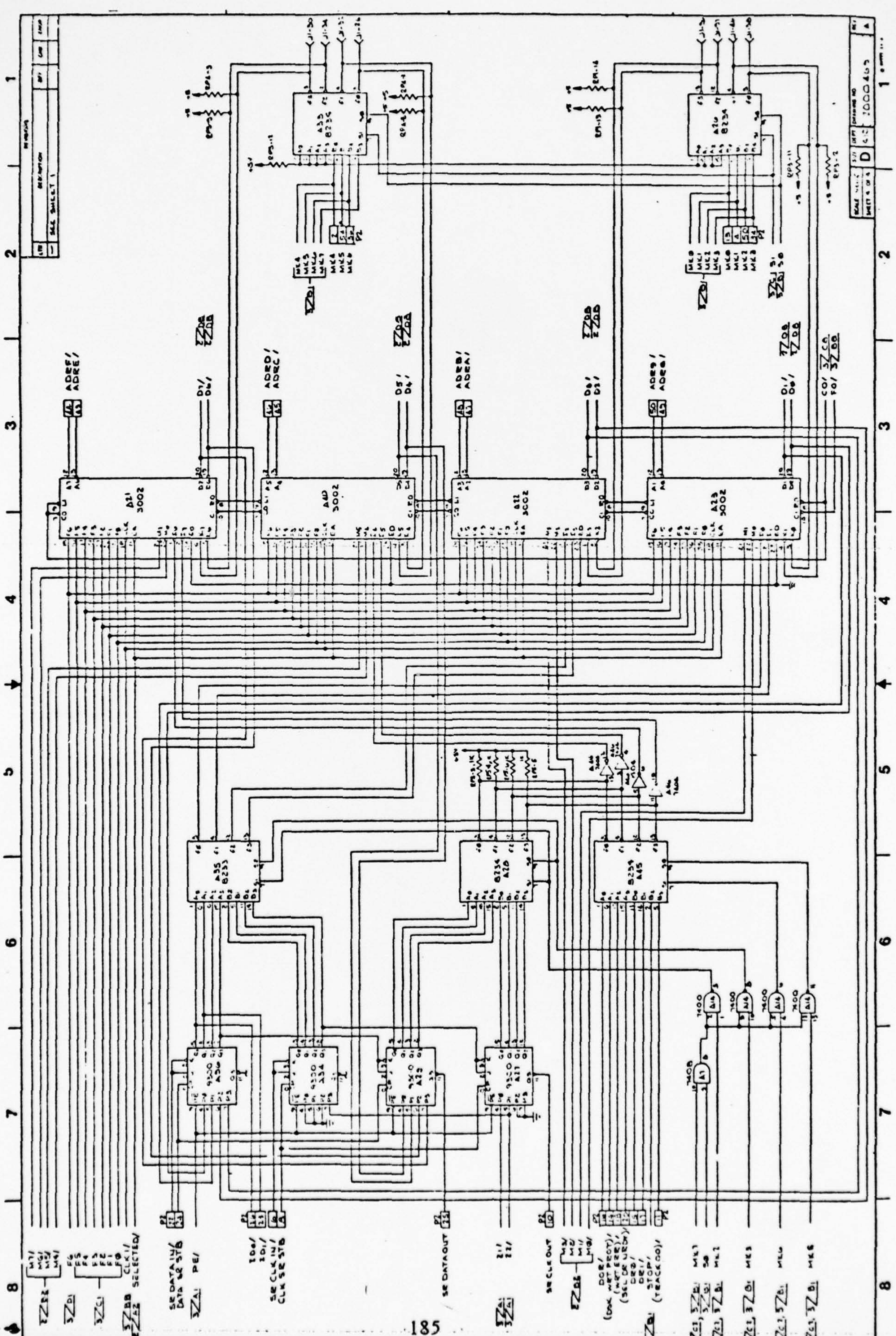


Fig. D-1. Channel Board Schematic


A
B
C
D
1
2
3
4
5
6
7
8
SELECTED/
GATE LOWER
MEM WRT
STB MEM IN
ADR 0/
ADR 7/
DAT 0/
DAT 7/
2000469



A4
3001
A10
3604
A11
A12
A16
3404
A17
A20
A5
8238
DEC OUT 7
DEC OUT 6
DEC OUT 5
DEC OUT 4
DEC OUT 1
DEC OUT 0
SR OUT
CLK 1/
MR
D
C
B
A
1
2
3
4
5
6
7
8
2000469



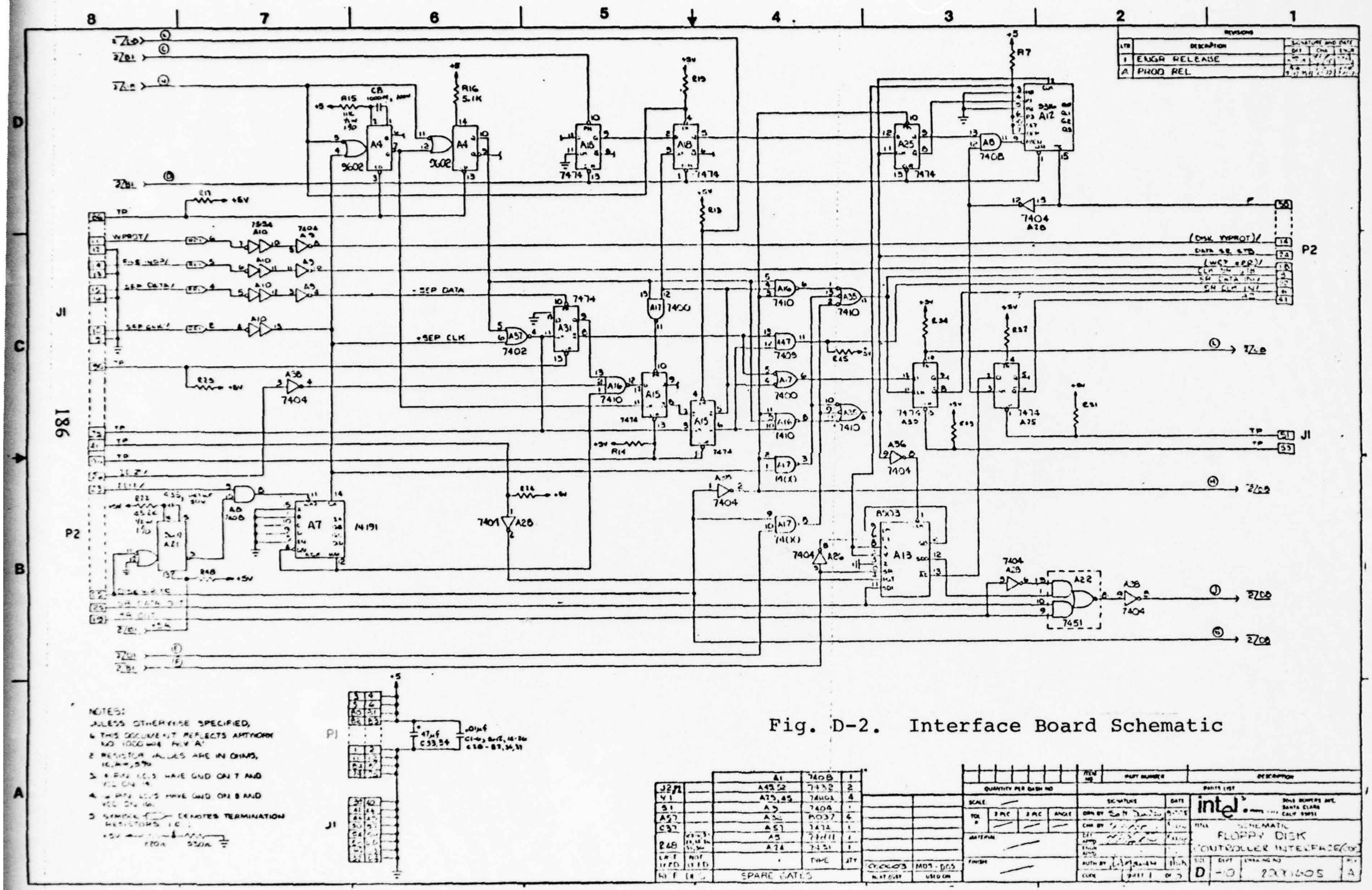
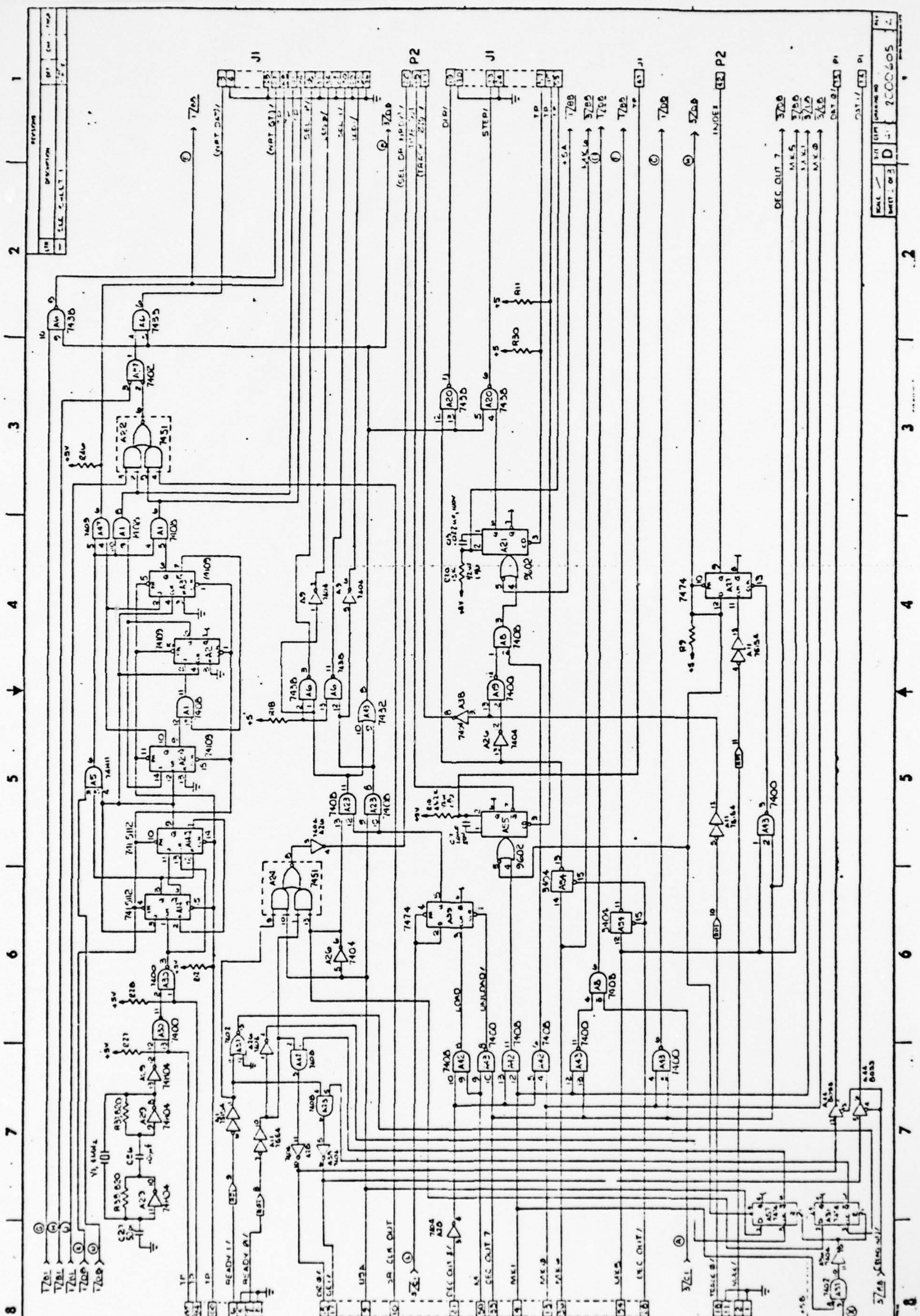
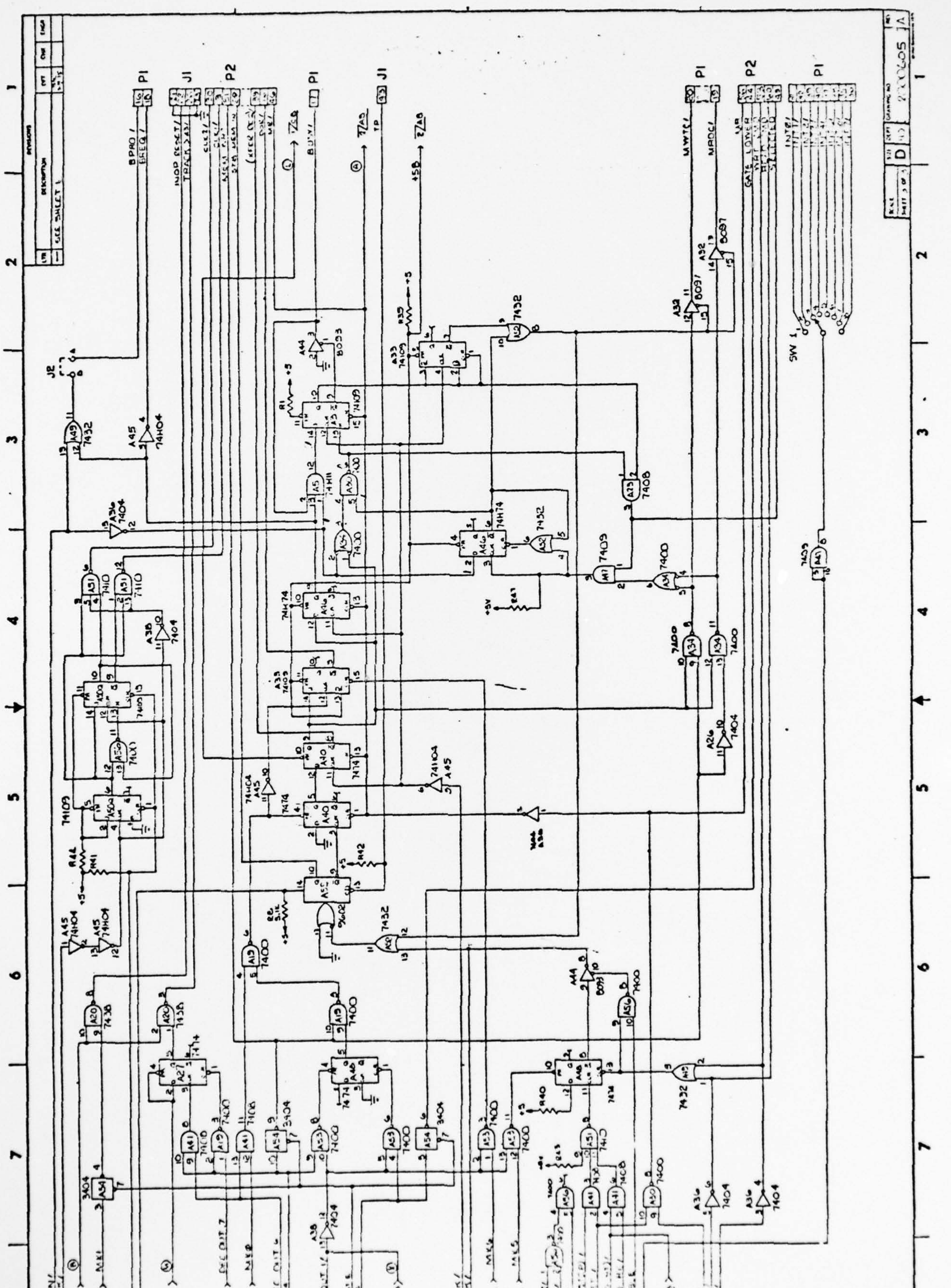

REVISIONS
1 ENGR RELEASE
A PROD REL
NOTES:
UNLESS OTHERWISE SPECIFIED,
1. THIS DOCUMENT REFLECTS ARTWORK NO. 1000404 REV A.
2. RESISTOR VALUES ARE IN OHMS, 1/4W, ±5%.
3. 14 PIN ICS HAVE GND ON 7 AND VCC ON 14.
4. 16 PIN ICS HAVE GND ON 8 AND VCC ON 16.
5. SYMBOL DENOTES TERMINATION RESISTORS I.C.
SPARE GATES
intel
SCHEMATIC
FLOPPY DISK
CONTROLLER INTERFACE
D
1001605
A
J1
P2
P1
-SEP DATA
+SEP CLK
WPROT/
7474
7404
7410
9602
74191
7451
A12
A13
A4
A7
A15
A18
A25
R7
R16 5.1K

Fig. D-2. Interface Board Schematic


J1
P2
J1
P2
P2
J1
J1
P2
7474
7400
7404
7408
7438
7451
9602
74LS112
2C00605
D


P1
J1
P2
J2
SW 1
7400
7404
7410
7432
7474
74109
74H74
8097
D
2A0605
A

Appendix E

DOS Controller - Pin Lists

Appendix E lists the 86-pin P1 bus connector assignments.

PIN P1	SIGNAL	TERMINATION	ORIGIN
25			
26			
27			
28			
29			
30			
31	CCLK/	CTLR/74H04	TTL
32			
33			
34			
35	INT6/		
36	INT7/		
37	INT4/		
38	INT5/		
39	INT2/		
40	INT3/		
41	INT0/	CPU/TTL	CTLR/TTL
42	INT1/		
43	ADRE/	Tri-State	Tri-State
44	ADRF/	Tri-State	Tri-State
45	ADRC/	Tri-State	Tri-State
46	ADRD/	Tri-State	Tri-State
47	ADRA/	Tri-State	Tri-State
48	ADRB/	Tri-State	Tri-State
49	ADR8/	Tri-State	Tri-State
50	ADR9/	Tri-State	Tri-State

PIN P1	SIGNAL	TERMINATION	ORIGIN
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13	BCLK/	CTLR/74H04	TTL
14	INIT/	CTLR/7400	CPU/TTL
15	BPRN/	CTLR	I
16	BPRO/	CTLR	I
17	BUSY/	CTLR	I
18	BREQ/	CTLR	I
19	MRDC/	Mem/TTL	CTLR/8097 Tri-State
20	MWTC/	Mem/TTL	CTLR/8097 Tri-State
21	IORC/	CTLR/7404	CPU/8095
22	IOWC/	CTLR/7404	CPU/8095 Tri-State
23	XACK/	CTLR/8093 Tri-State	CTLR/8093/7432 Tri-State
24			

PIN P1	SIGNAL	TERMINATION	ORIGIN
51	ADR6/	Tri-State	Tri-State
52	ADR7/	Tri-State	Tri-State
53	ADR4/	Tri-State	Tri-State
54	ADR5/	Tri-State	Tri-State
55	ADR2/	Tri-State	Tri-State
56	ADR3/	Tri-State	Tri-State
57	ADR0/	Tri-State	Tri-State
58	ADR1l	Tri-State	Tri-State
59	DATAE/	Tri-State	Tri-State
60	DATAF/	Tri-State	Tri-State
61	DATAC/	Tri-State	Tri-State
62	DATAD/	Tri-State	Tri-State
63	DATAA/	Tri-State	Tri-State
64	DATAB/	Tri-State	Tri-State
65	DATA8/	Tri-State	Tri-State
66	DATA9/	Tri-State	Tri-State
67	DATA6/	Tri-State	Tri-State
68	DATA7/	Tri-State	Tri-State
69	DATA4/	Tri-State	Tri-State
70	DATA5/	Tri-State	Tri-State
71	DATA2/	Tri-State	Tri-State
72	DATA3/	Tri-State	Tri-State
73	DATA0/	Tri-State	Tri-State
74	DATA1/	Tri-State	Tri-State
75			
76			
77			
78			

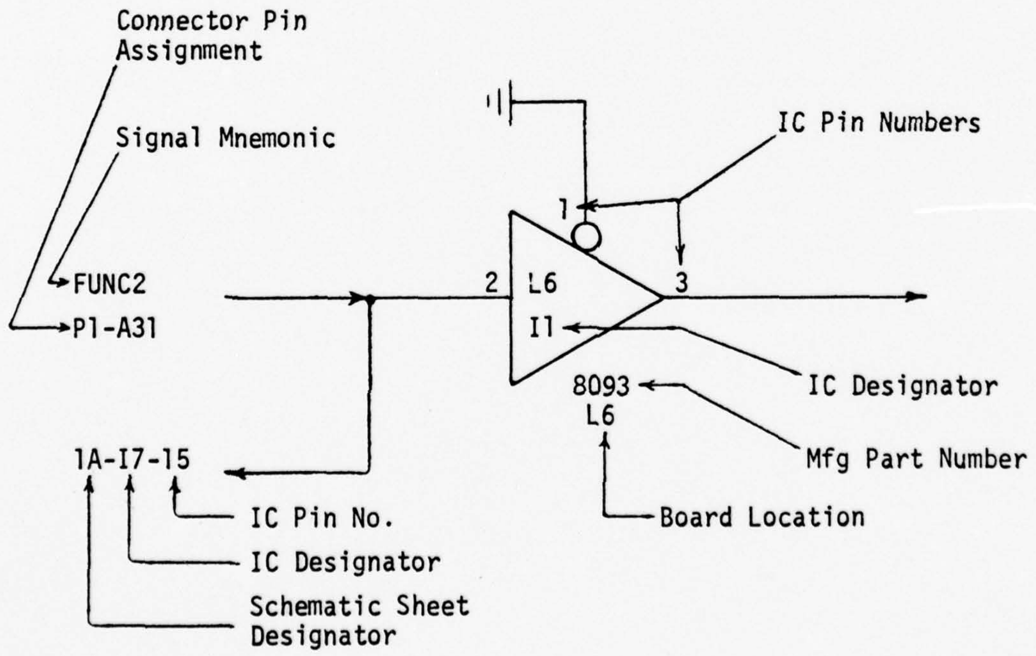
PIN P1	SIGNAL	TERMINATION	ORIGIN
79			
80			
81			
82			
83			
84			
85			
86			

APPENDIX F

Interface Design Schematics

The logic used to implement the I/O, DMA, and interrupt functions for the entire interface design is carefully laid out in the schematics of this appendix.

An example which explains the information contained on each logic gate follows:



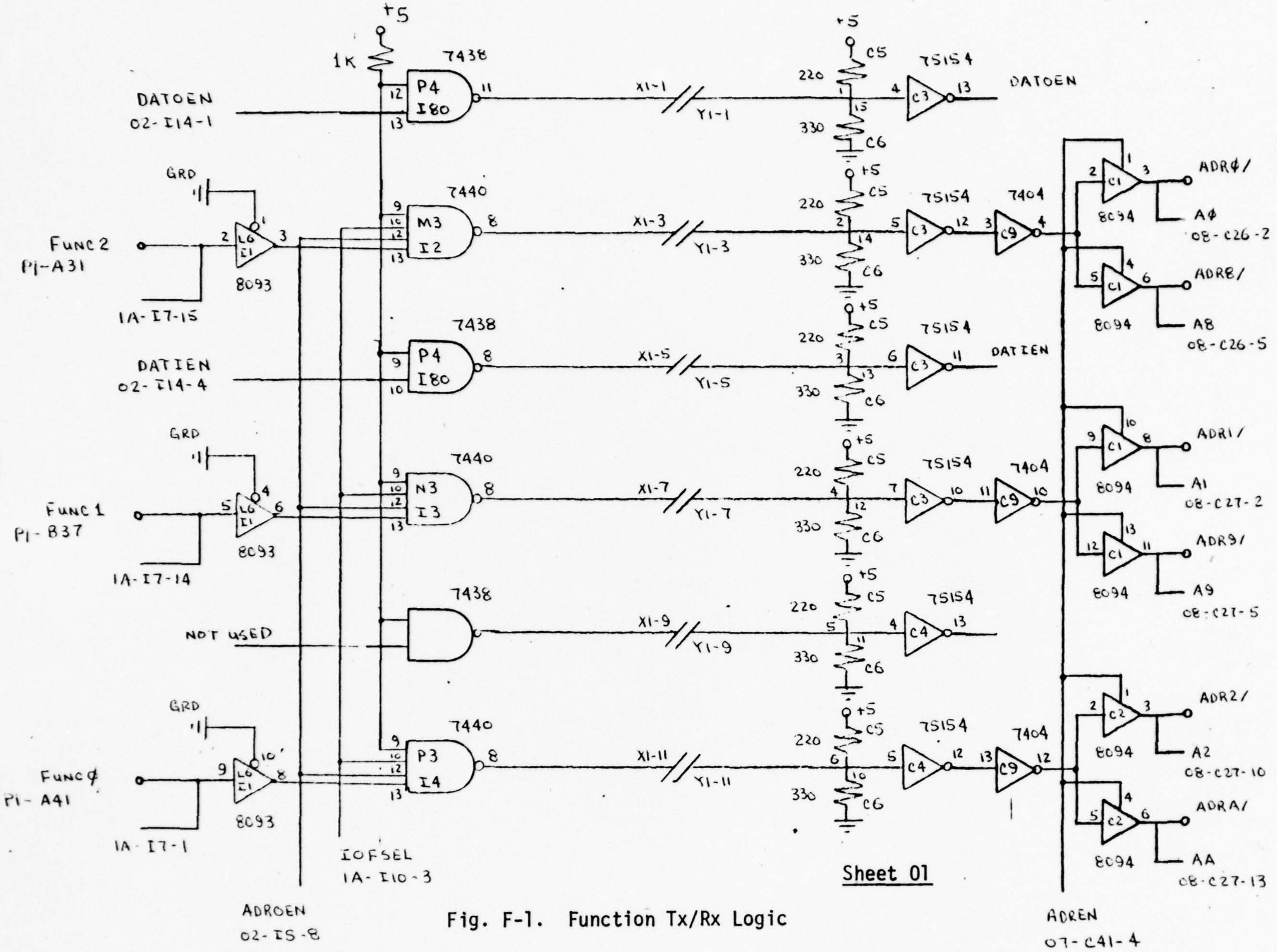


Fig. F-1. Function Tx/Rx Logic

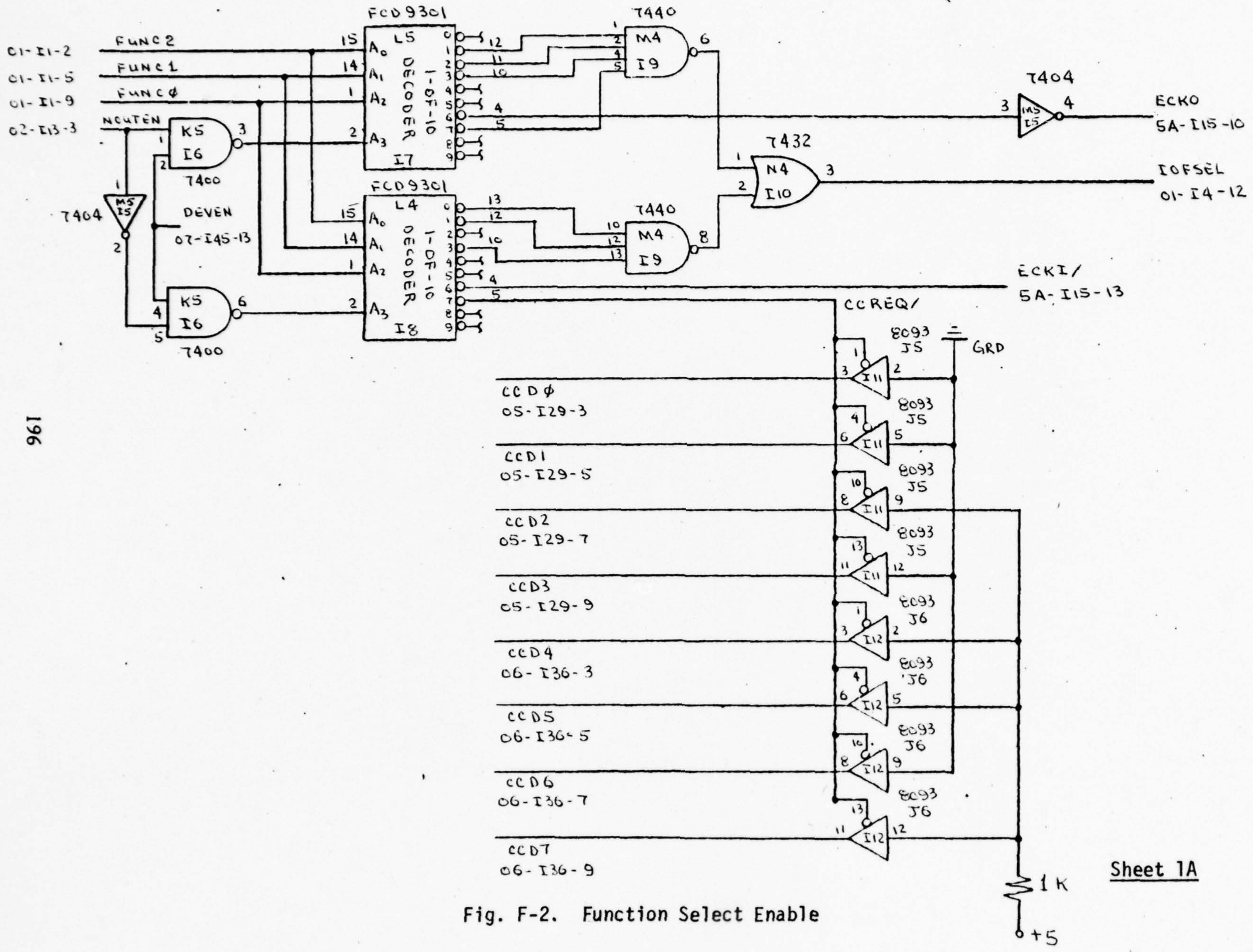
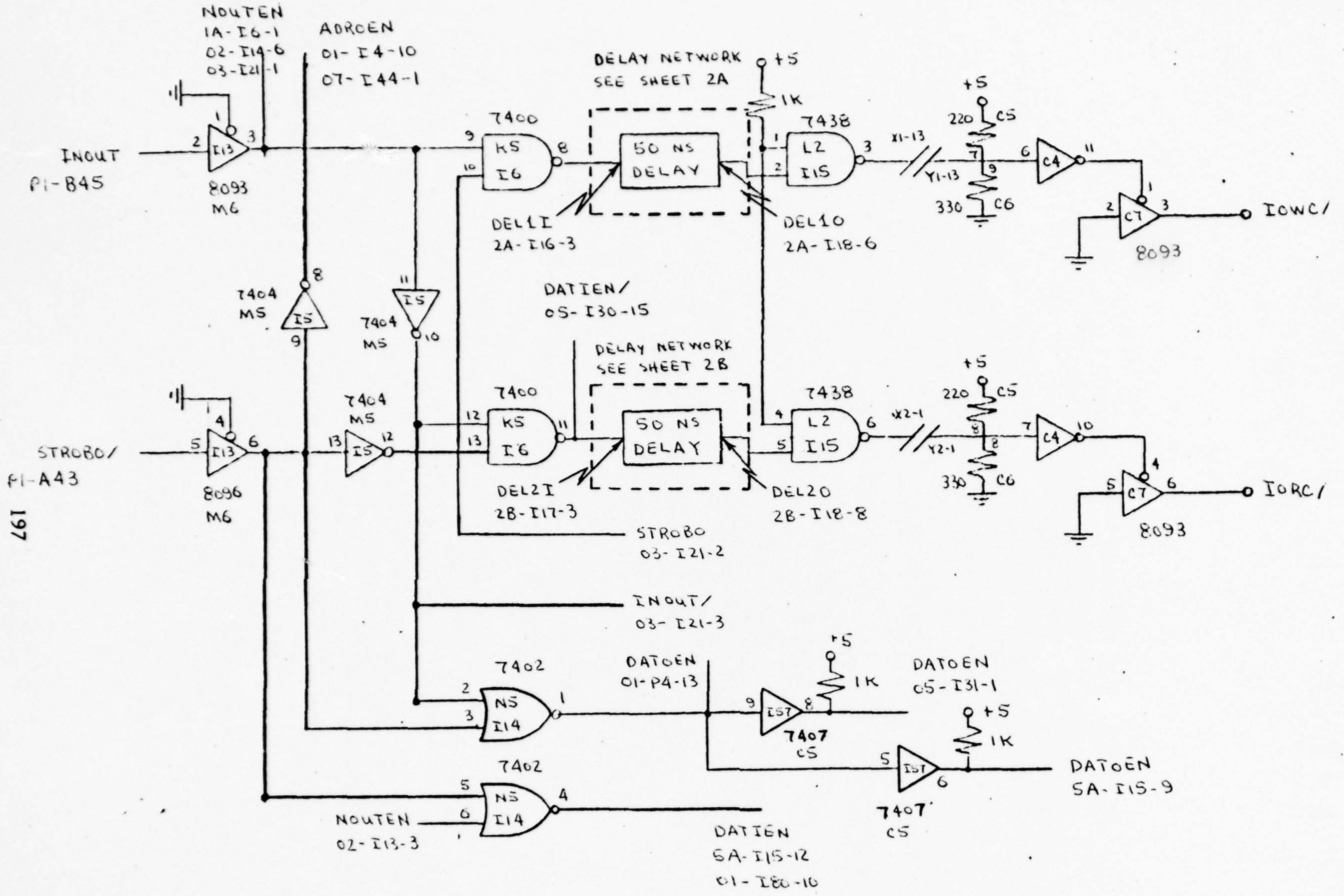
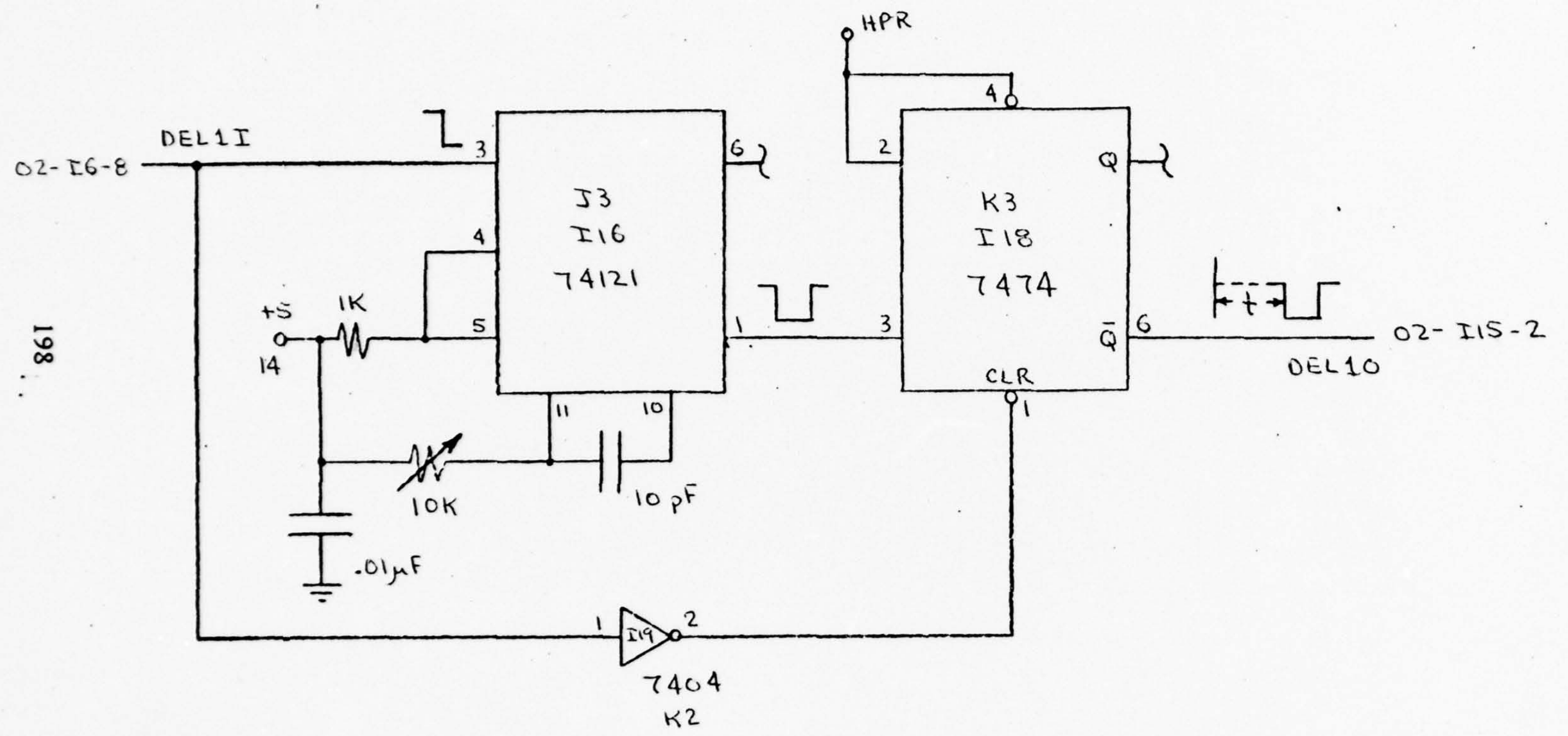


Fig. F-2. Function Select Enable



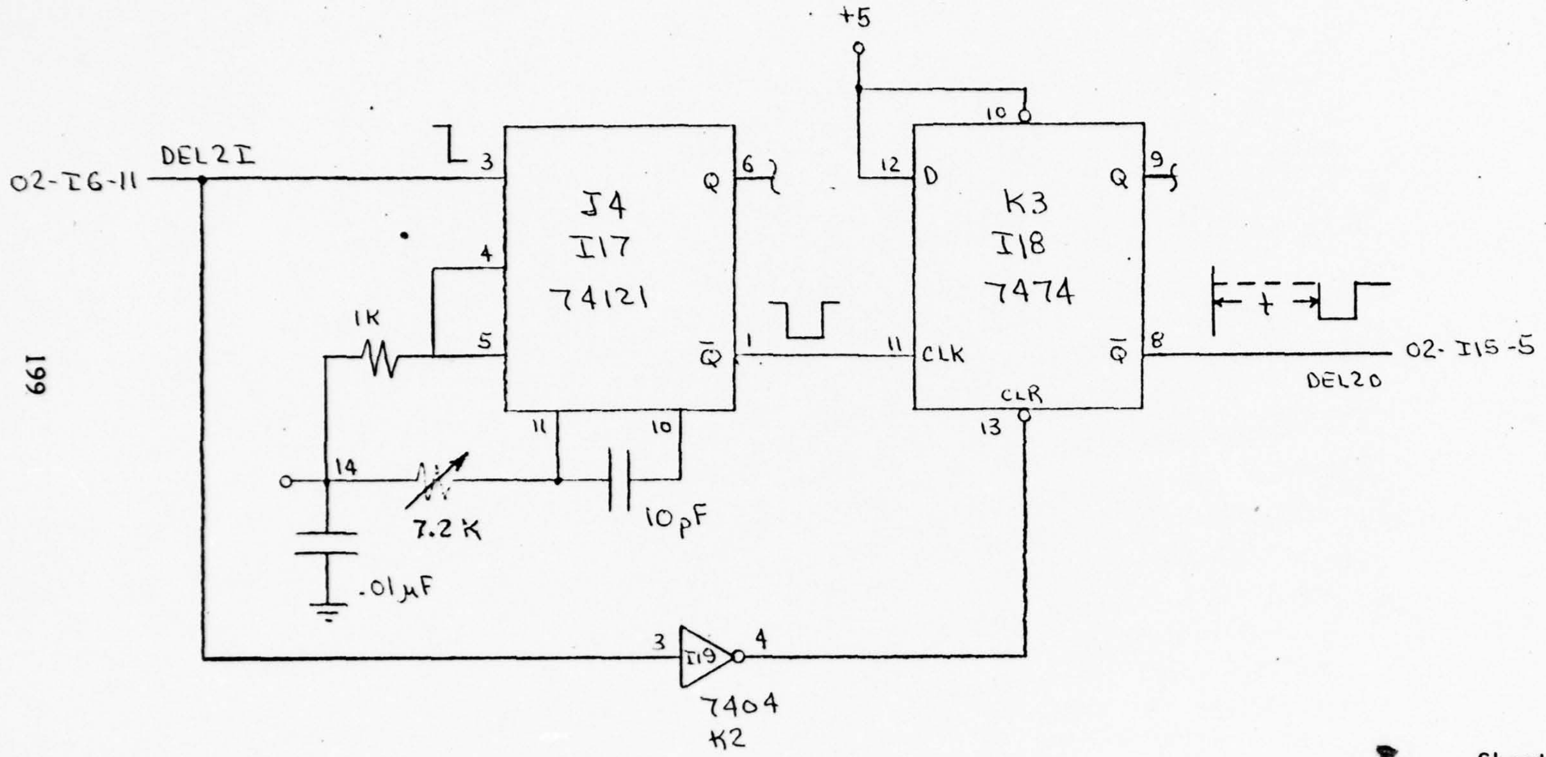
Sheet 02

Fig. F-3. Read/Write Control



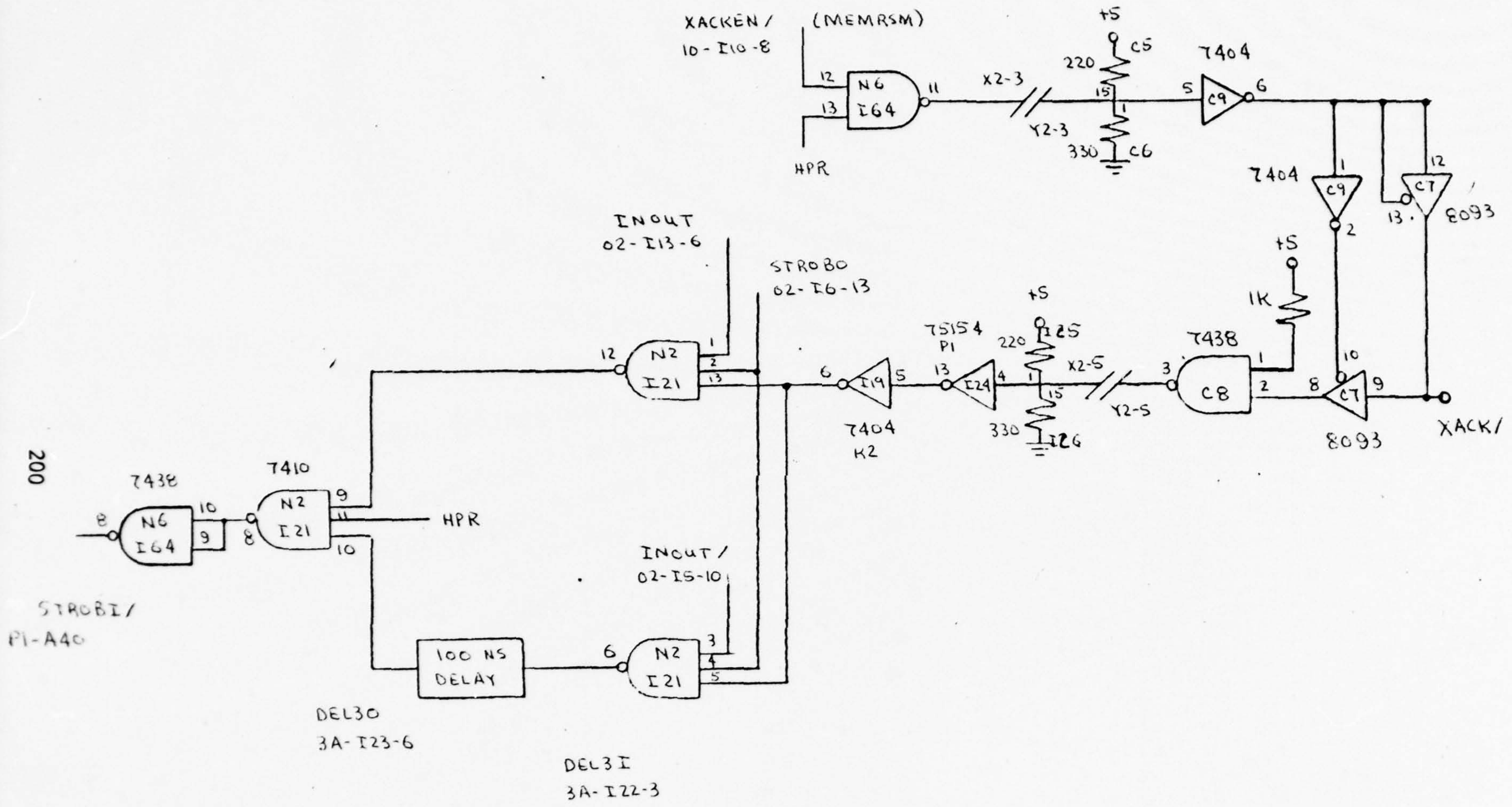
Sheet 2A

Fig. F-4. IOWC/ Delay Logic



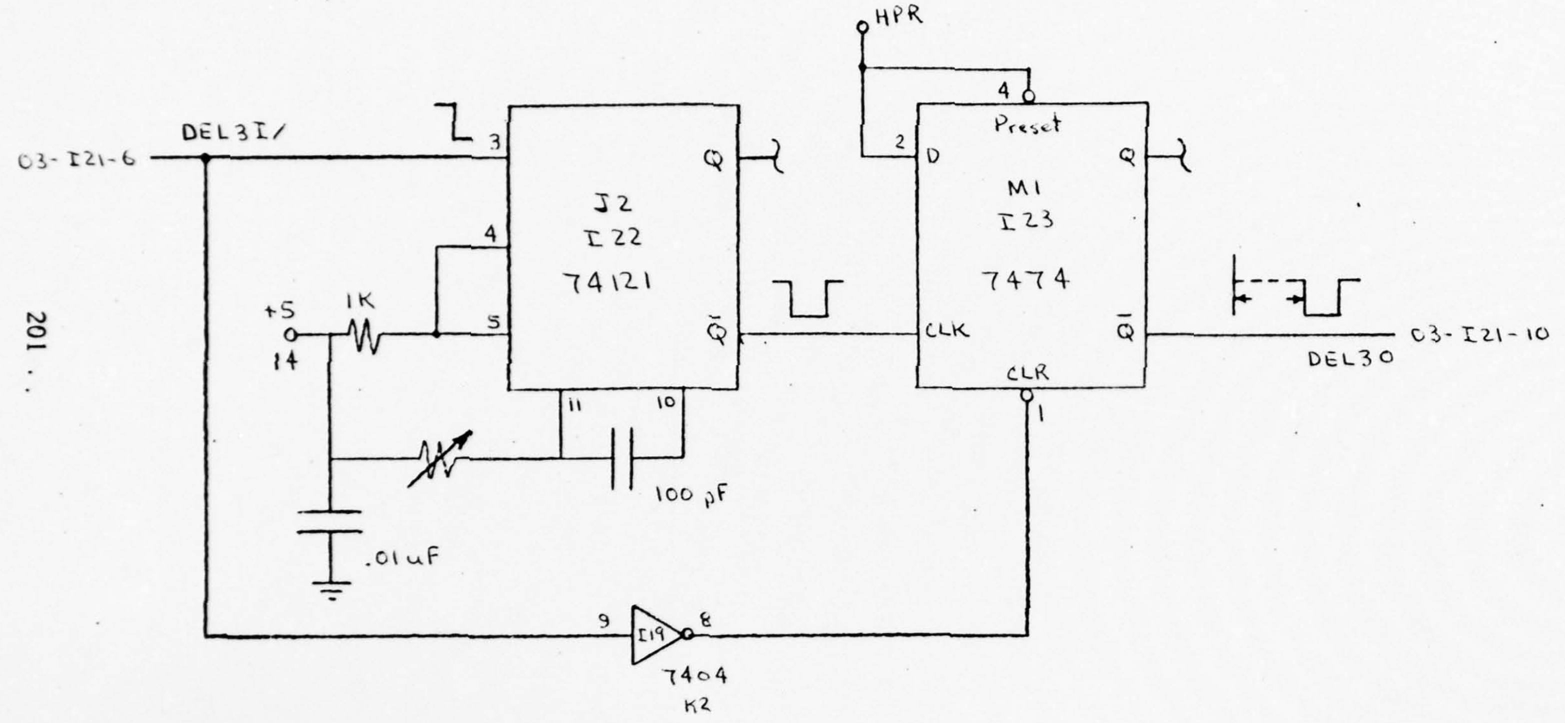
Sheet 2B

Fig. F-5. IORC/ Delay Logic



Sheet 03

Fig. F-6. Transfer Acknowledge Logic



Sheet 3A

Fig. F-7. STROBI/ Delay Logic

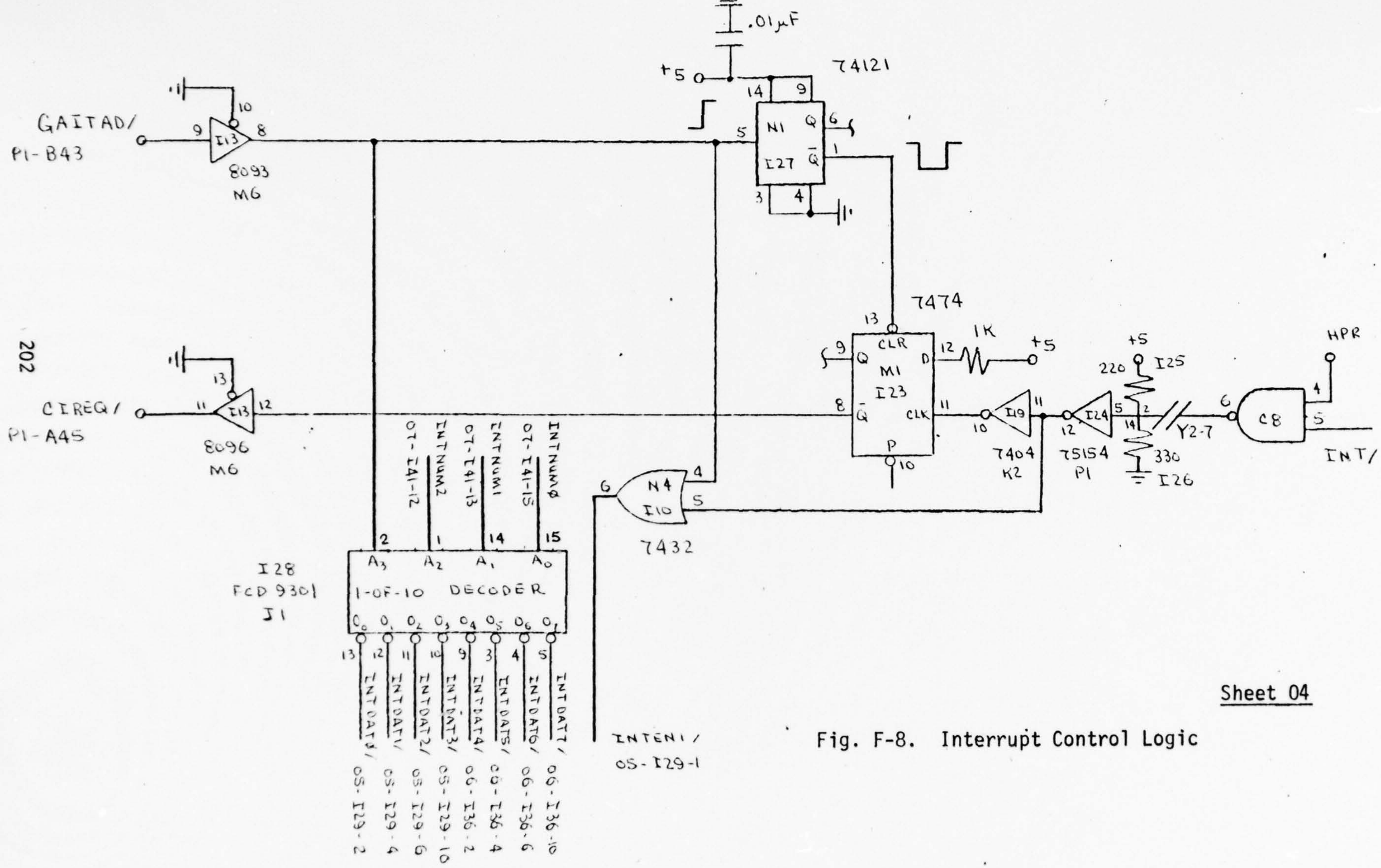
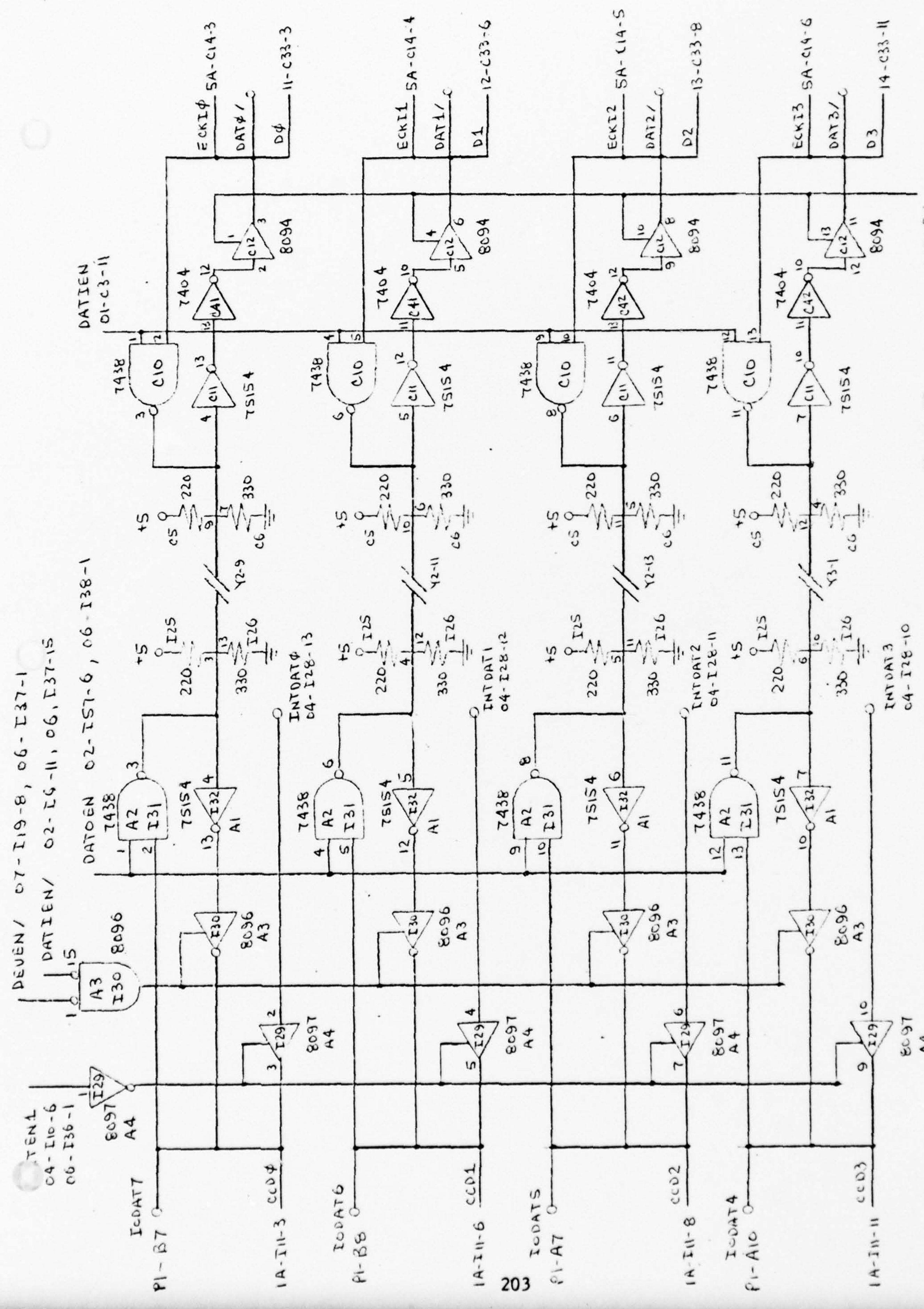


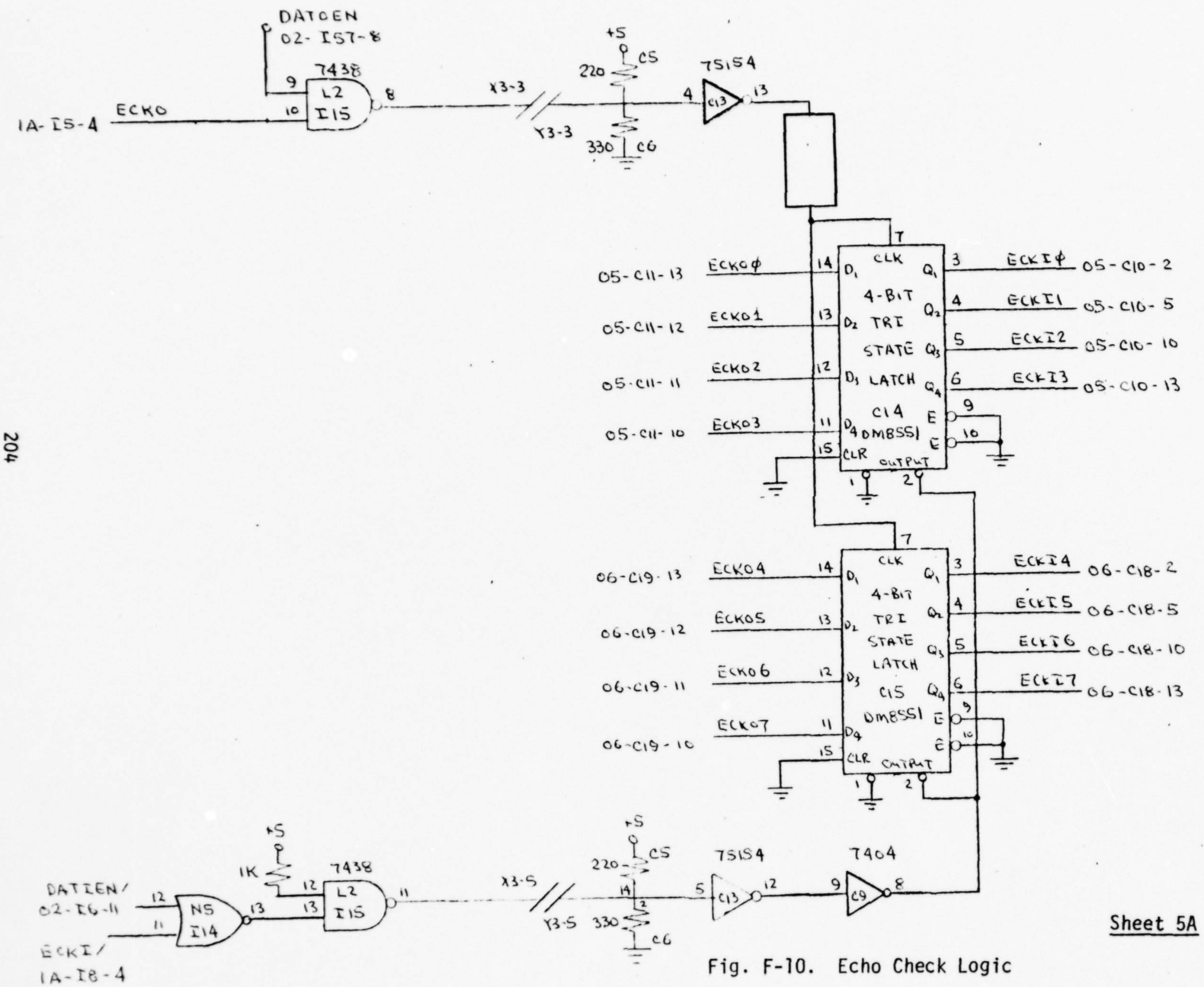
Fig. F-8. Interrupt Control Logic

Sheet 04



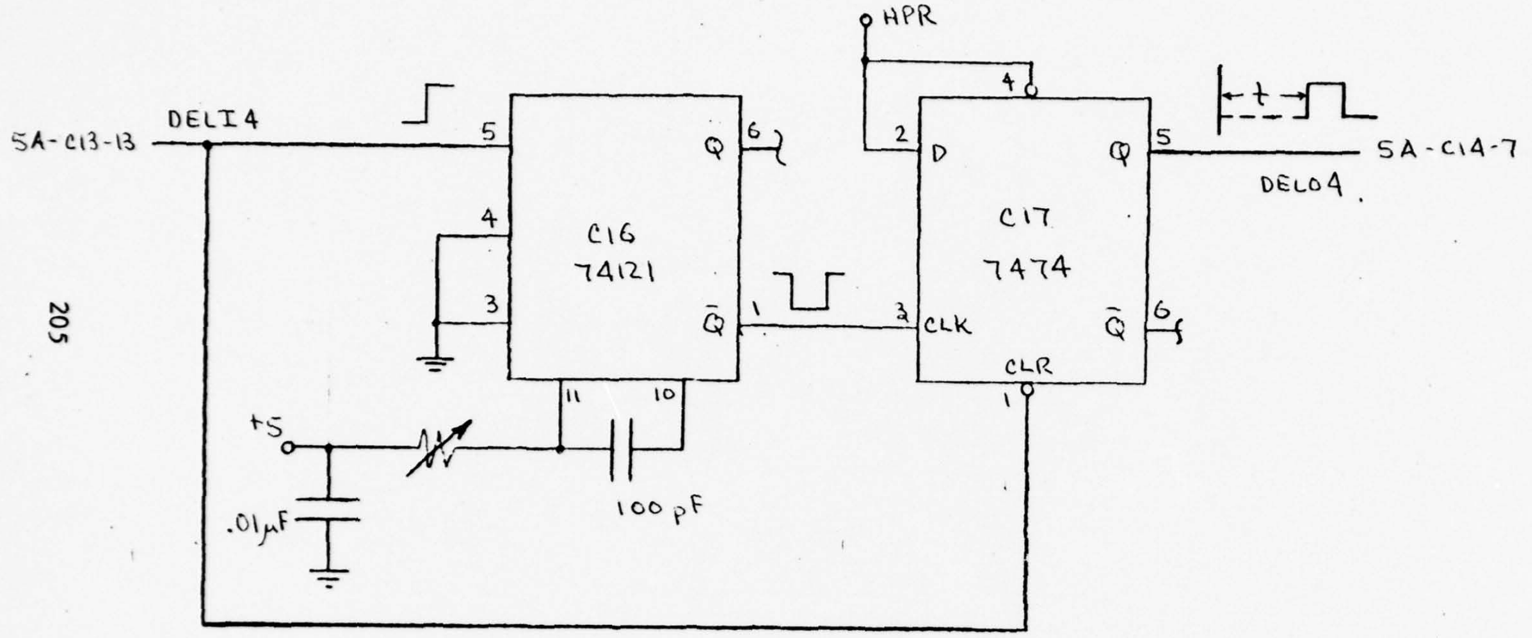
Sheet 05

Fig. F-9. I/O Data Select Logic



Sheet 5A

Fig. F-10. Echo Check Logic



Sheet 5B

Fig. F-11. ECKO Delay Logic

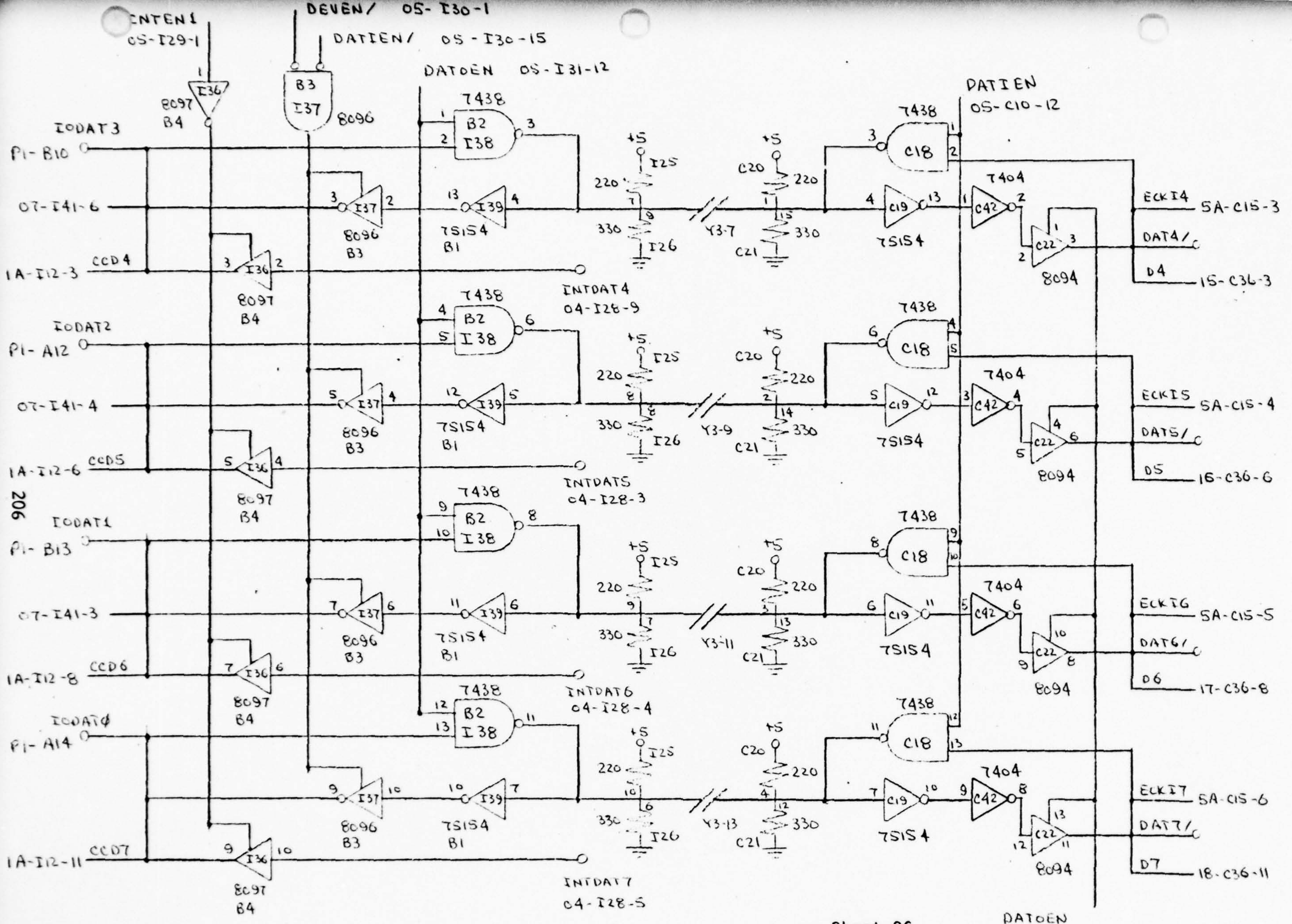


Fig. F-12. I/O Data Select Logic

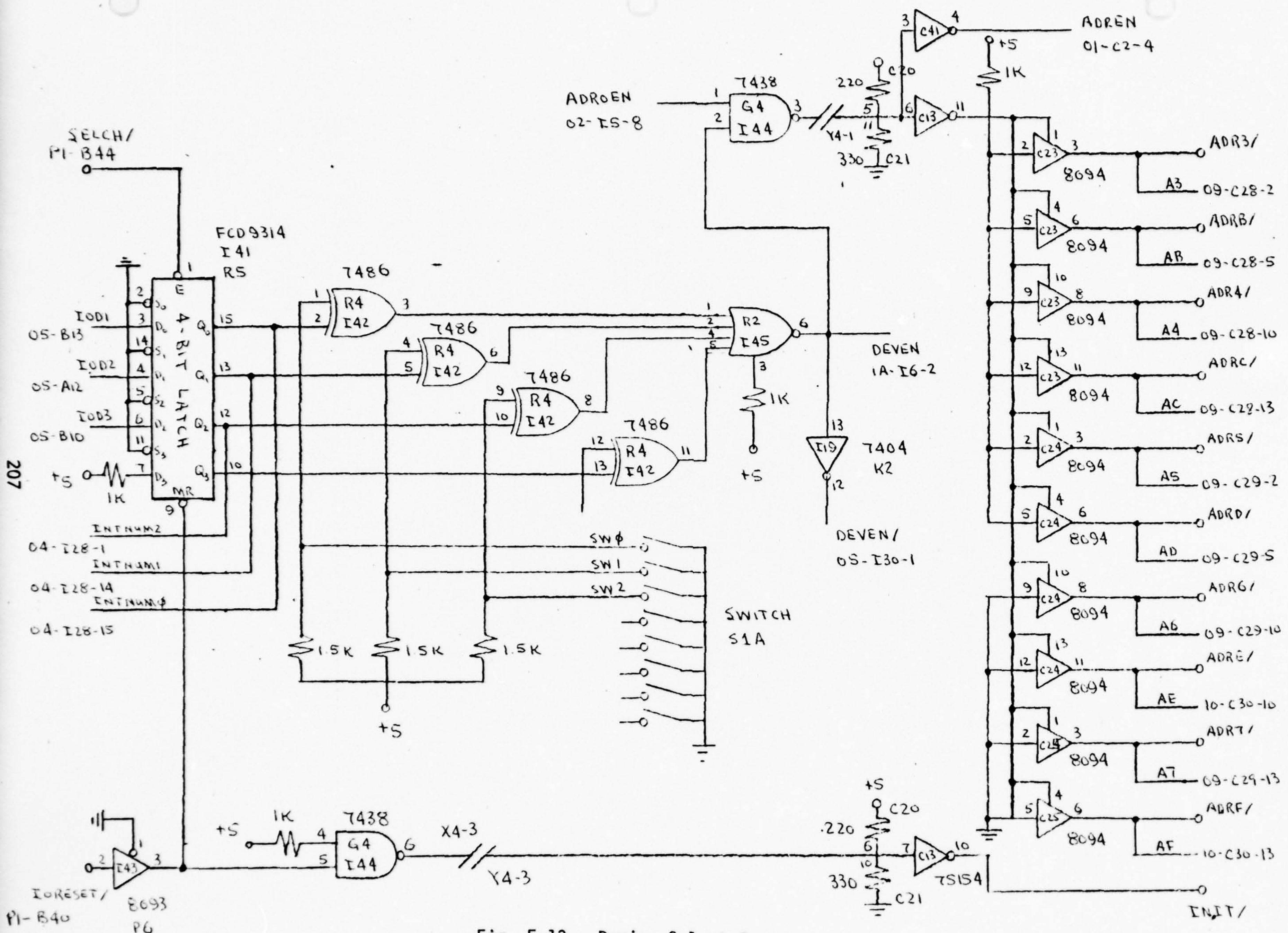
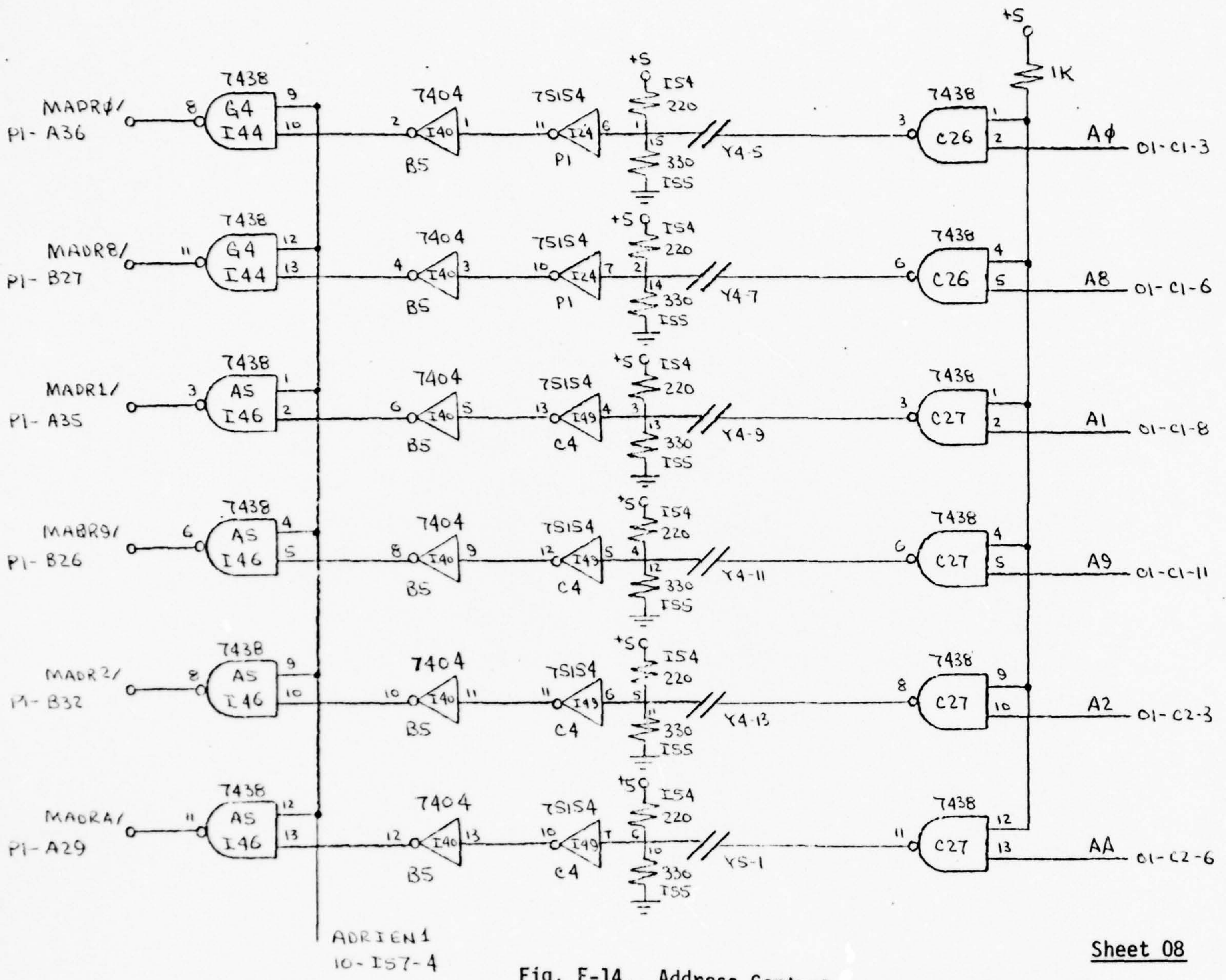


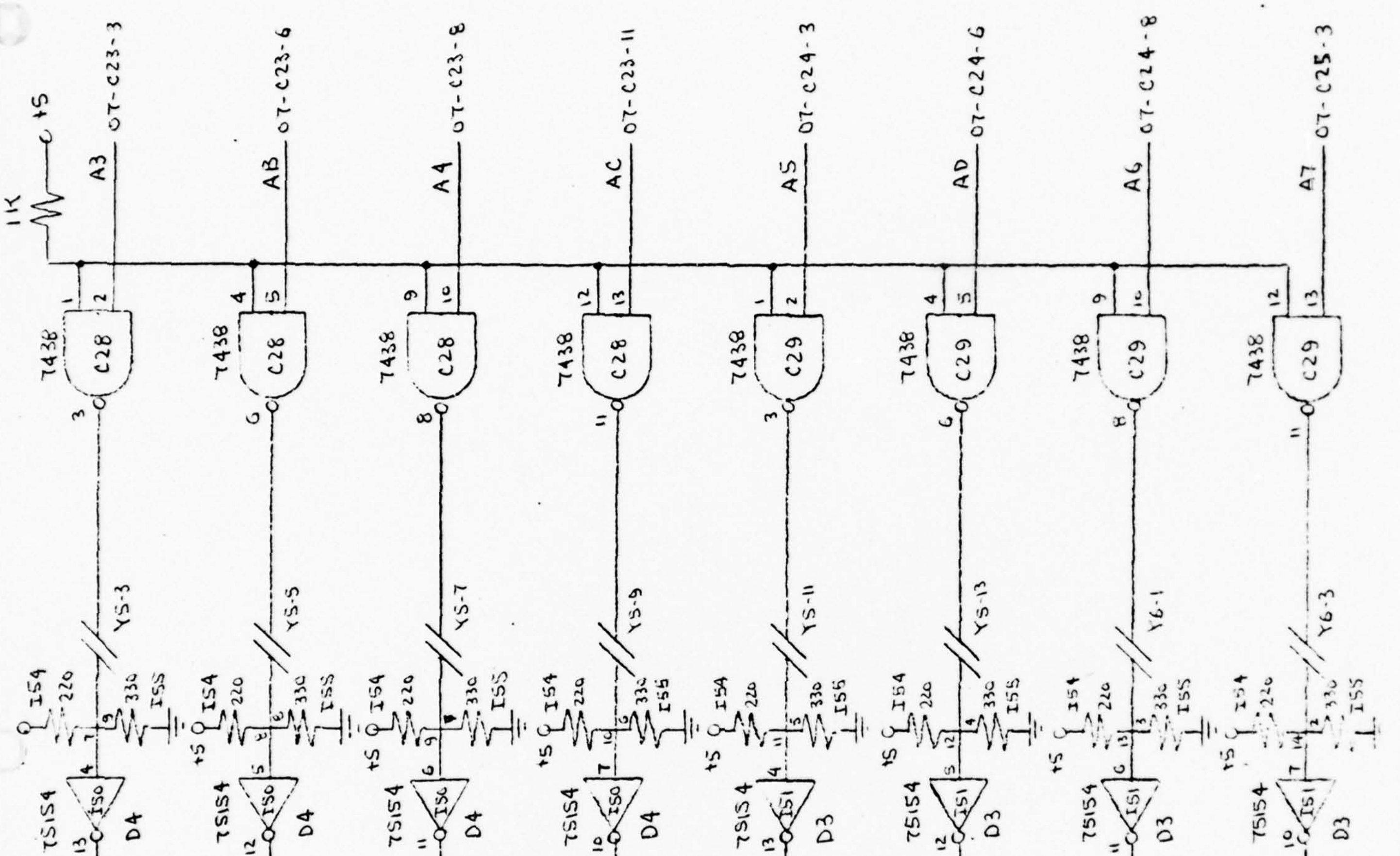
Fig. F-13. Device Select Control

Sheet 07



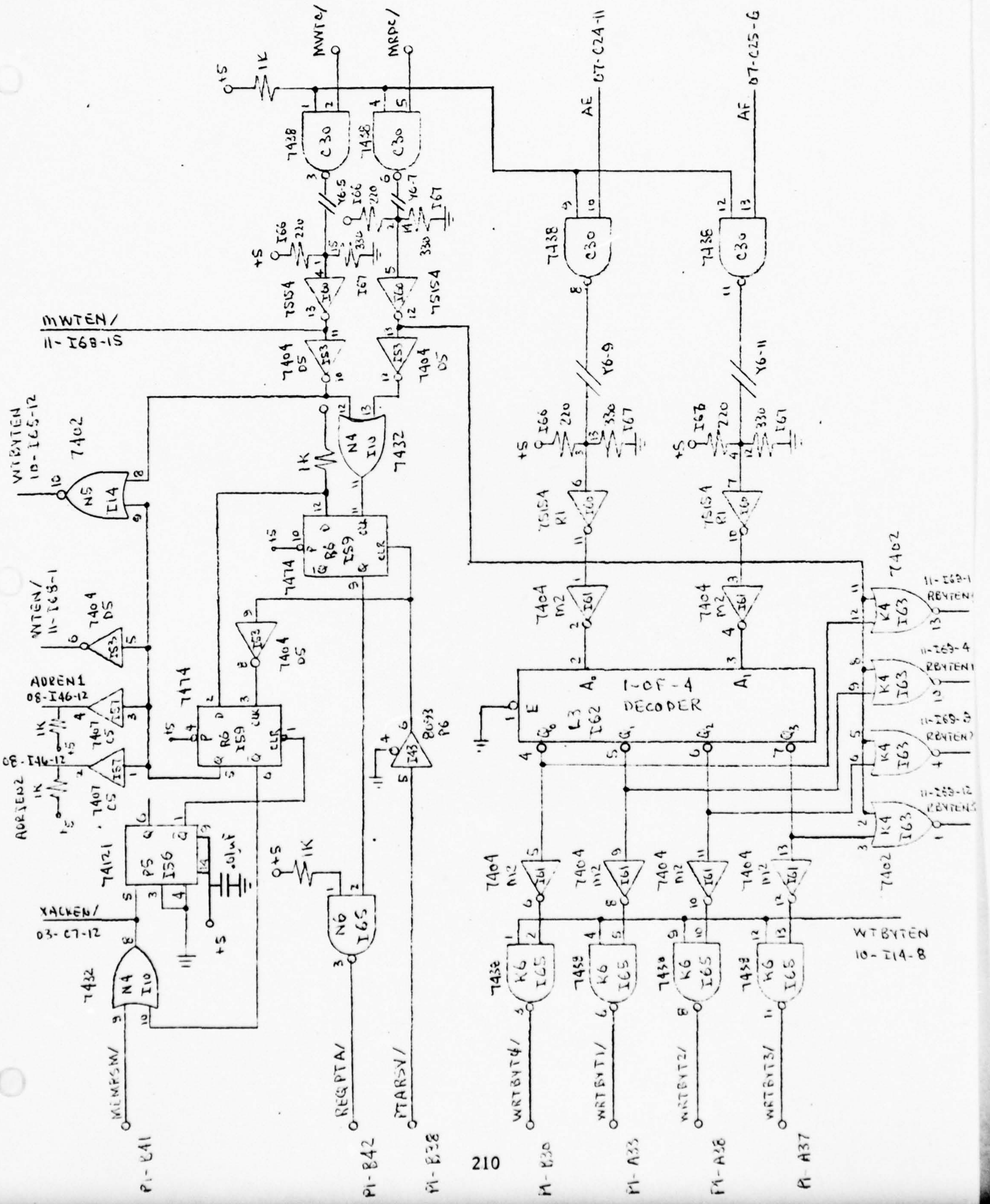
Sheet 08

Fig. F-14. Address Control Logic

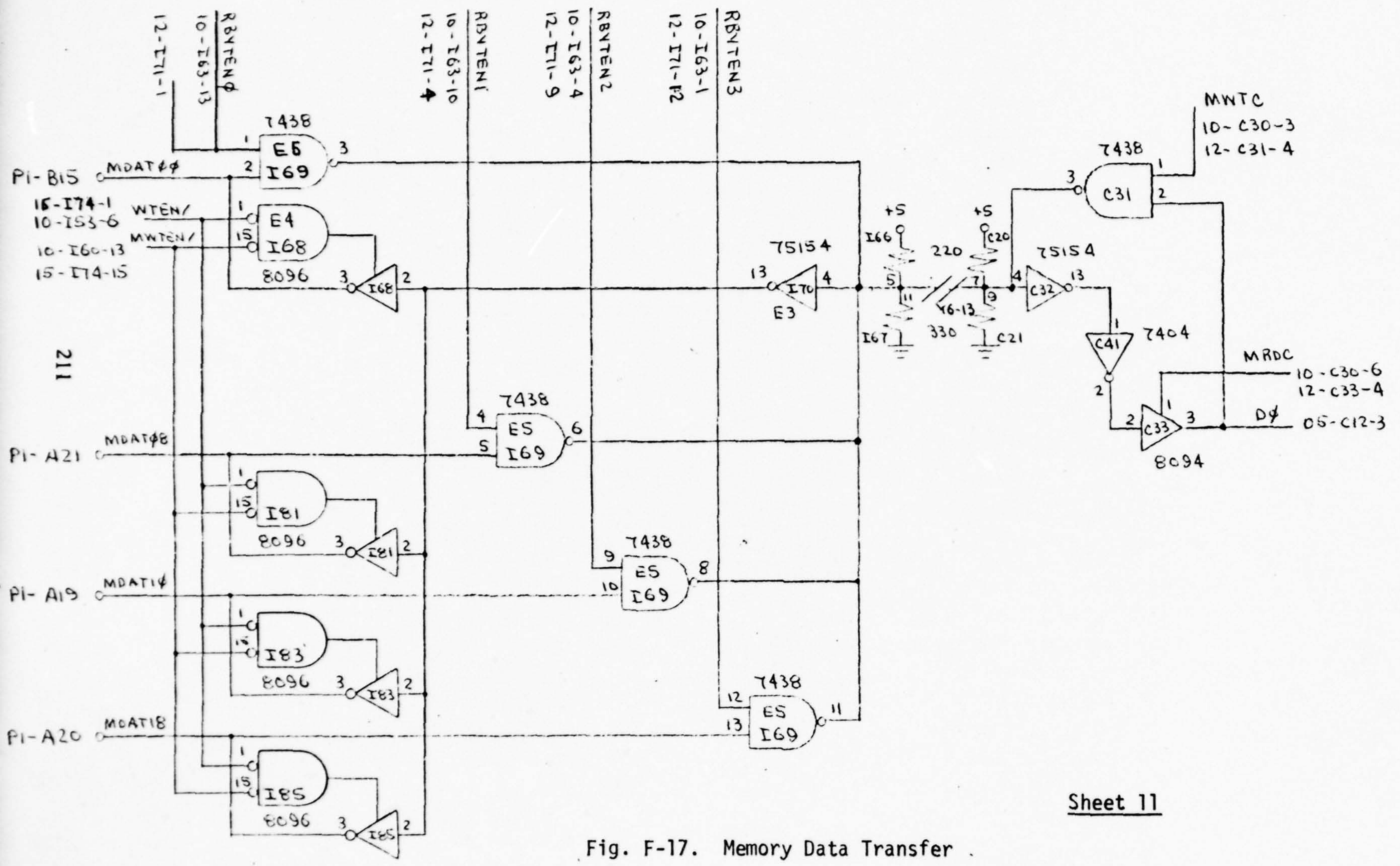


Sheet 09

Fig. F-15. Address Control Logic

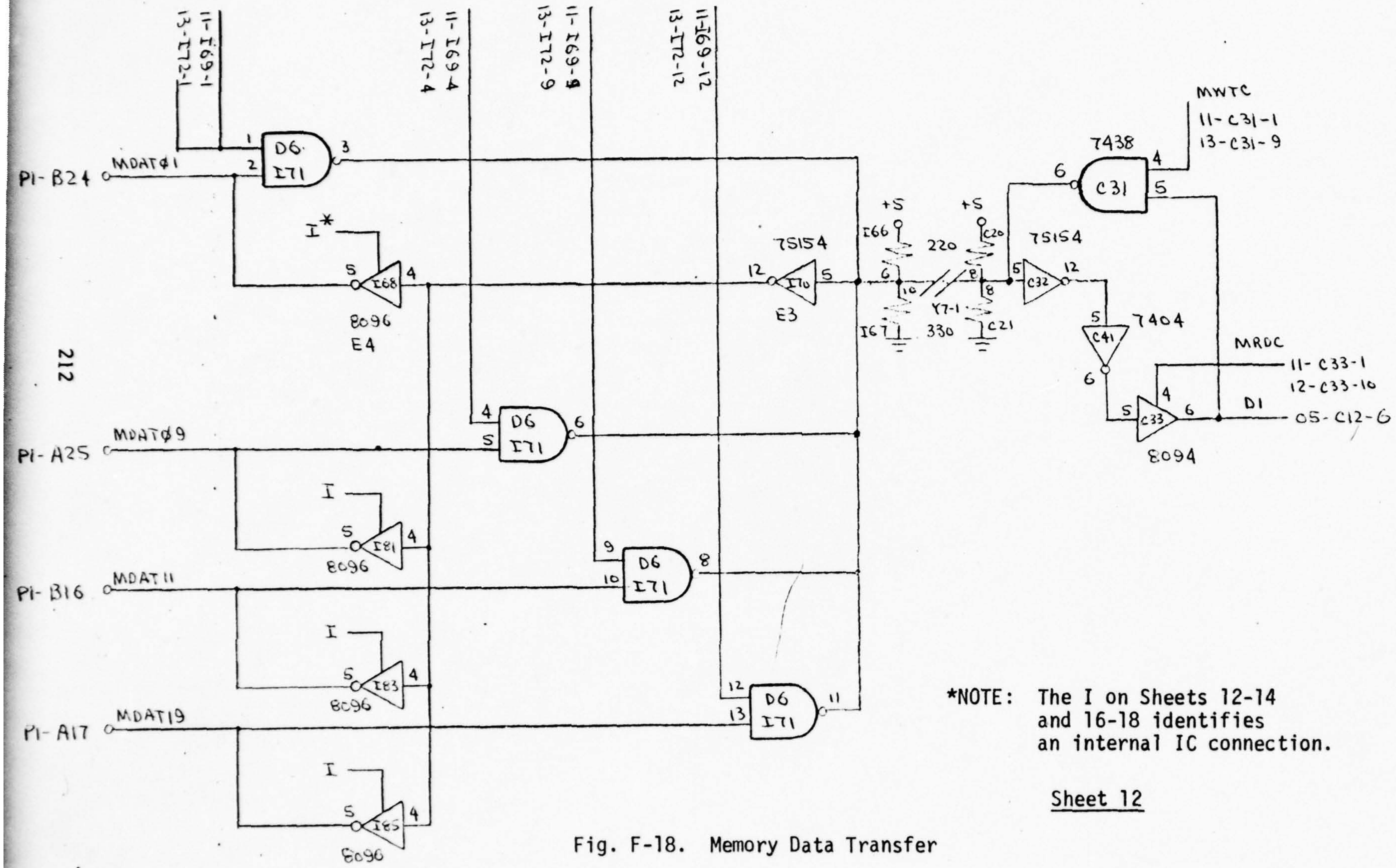


MWTC/
MRDC/
AE 07-C24-11
AF 07-C25-G
MWTEN/ 11-I68-15
WTBYTEN 10-I65-12
WTEN/ 11-I68-1
ADREN1 08-I46-12
ADRSEN2 08-I46-12
XACKEN/ 03-C7-12
1-OF-4 DECODER
RBYTEN 11-I69-1
RBYTEN1 11-I69-4
RBYTEN2 11-I69-9
RBYTEN3 11-I69-12
WTBYTEN 10-I14-8
MEMRSM/ P1-B41
REQPTA/ P1-B42
MTARSV/ P1-B38
WRTBYT4/ P1-B30
WRTBYT1/ P1-A33
WRTBYT2/ P1-A38
WRTBYT3/ P1-A37
7438
7404
75154
7432
7402
7474
74121
8093



Sheet 11

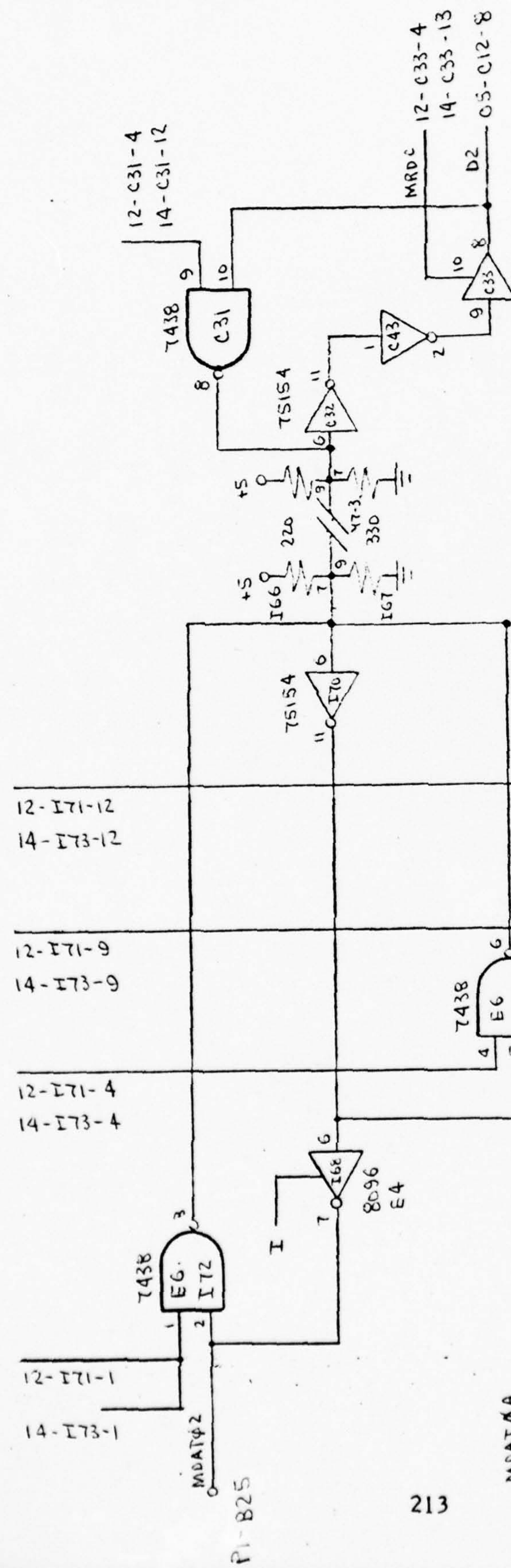
Fig. F-17. Memory Data Transfer



*NOTE: The I on Sheets 12-14 and 16-18 identifies an internal IC connection.

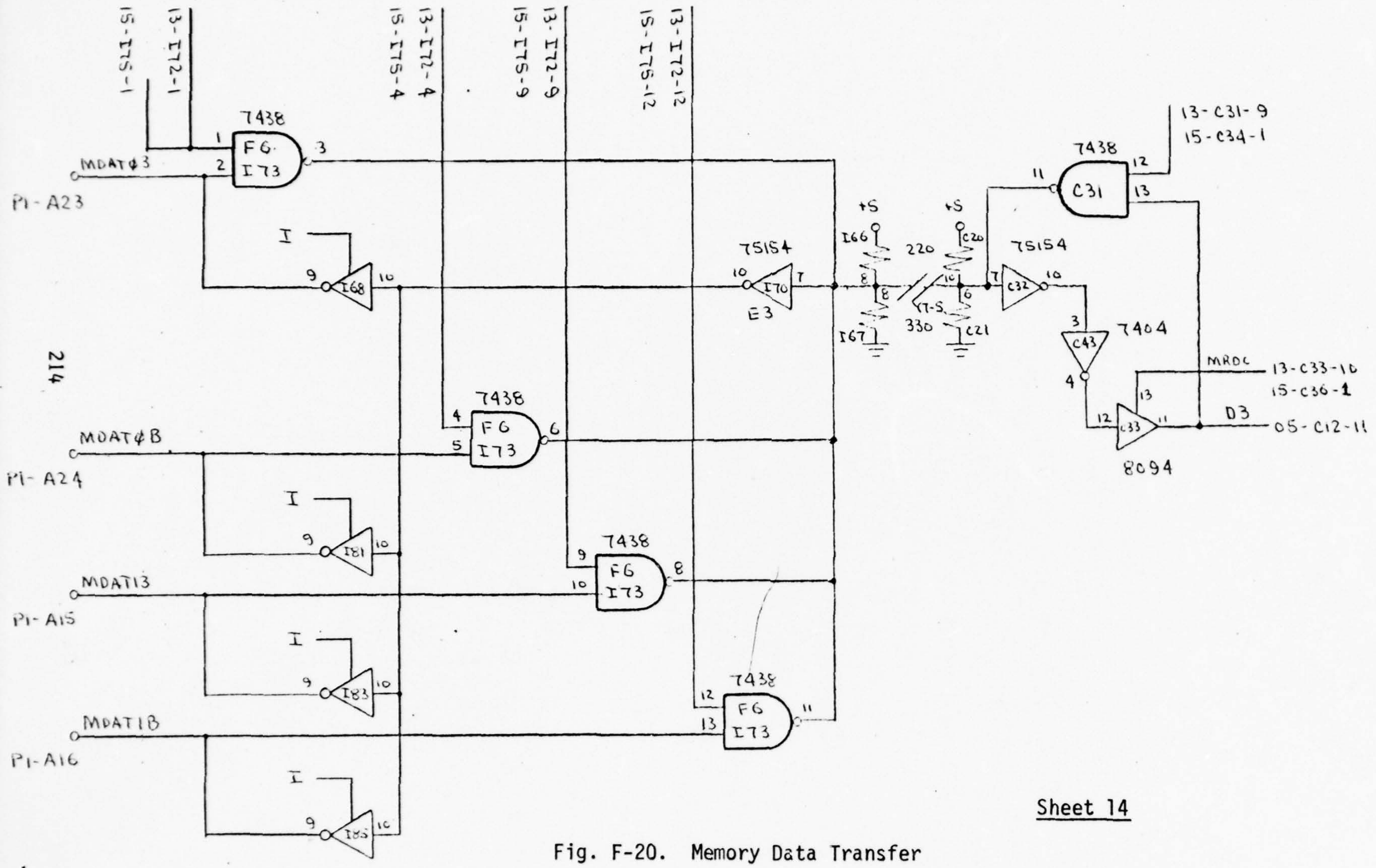
Sheet 12

Fig. F-18. Memory Data Transfer



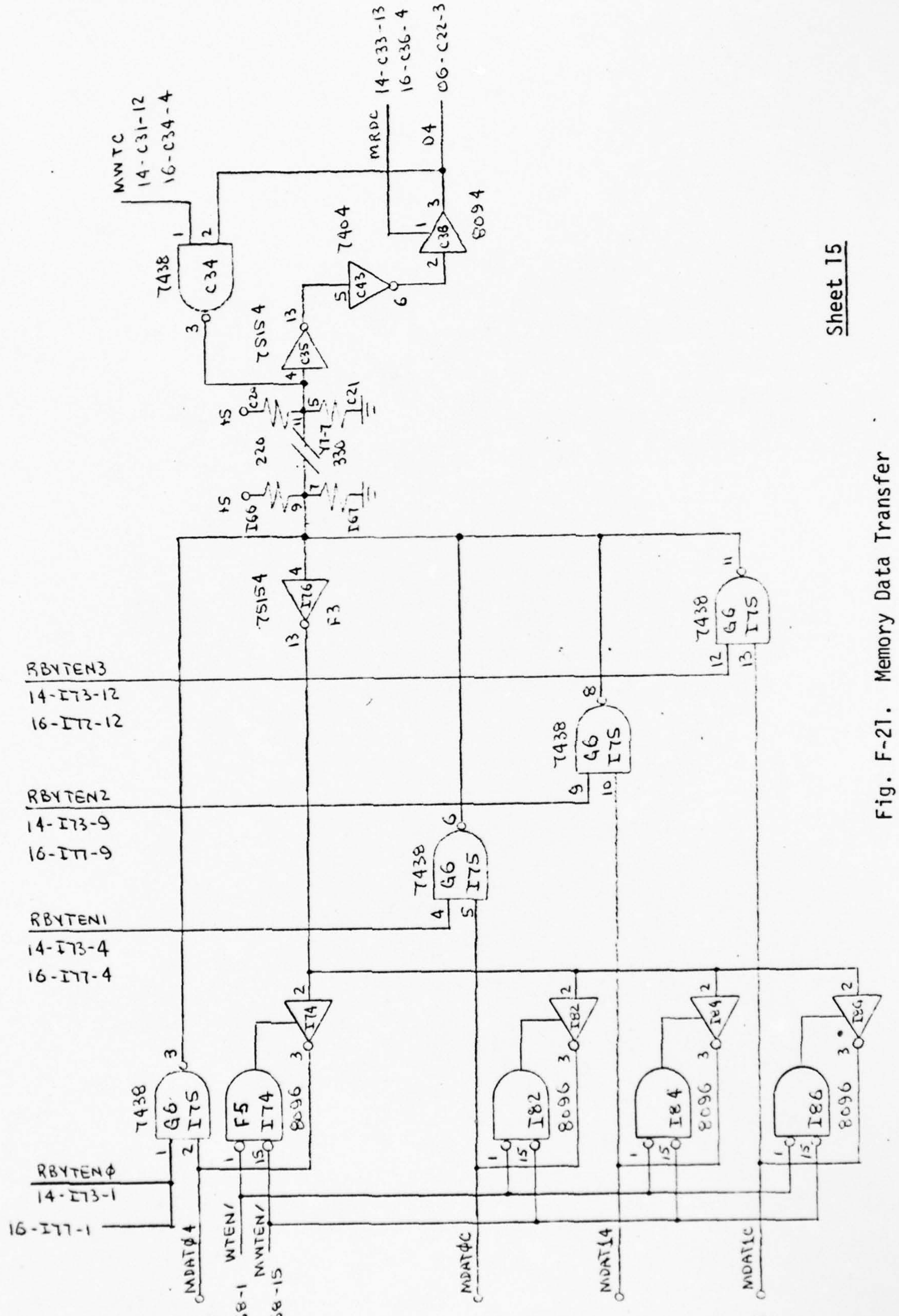
Sheet 13

Fig. F-19. Memory Data Transfer



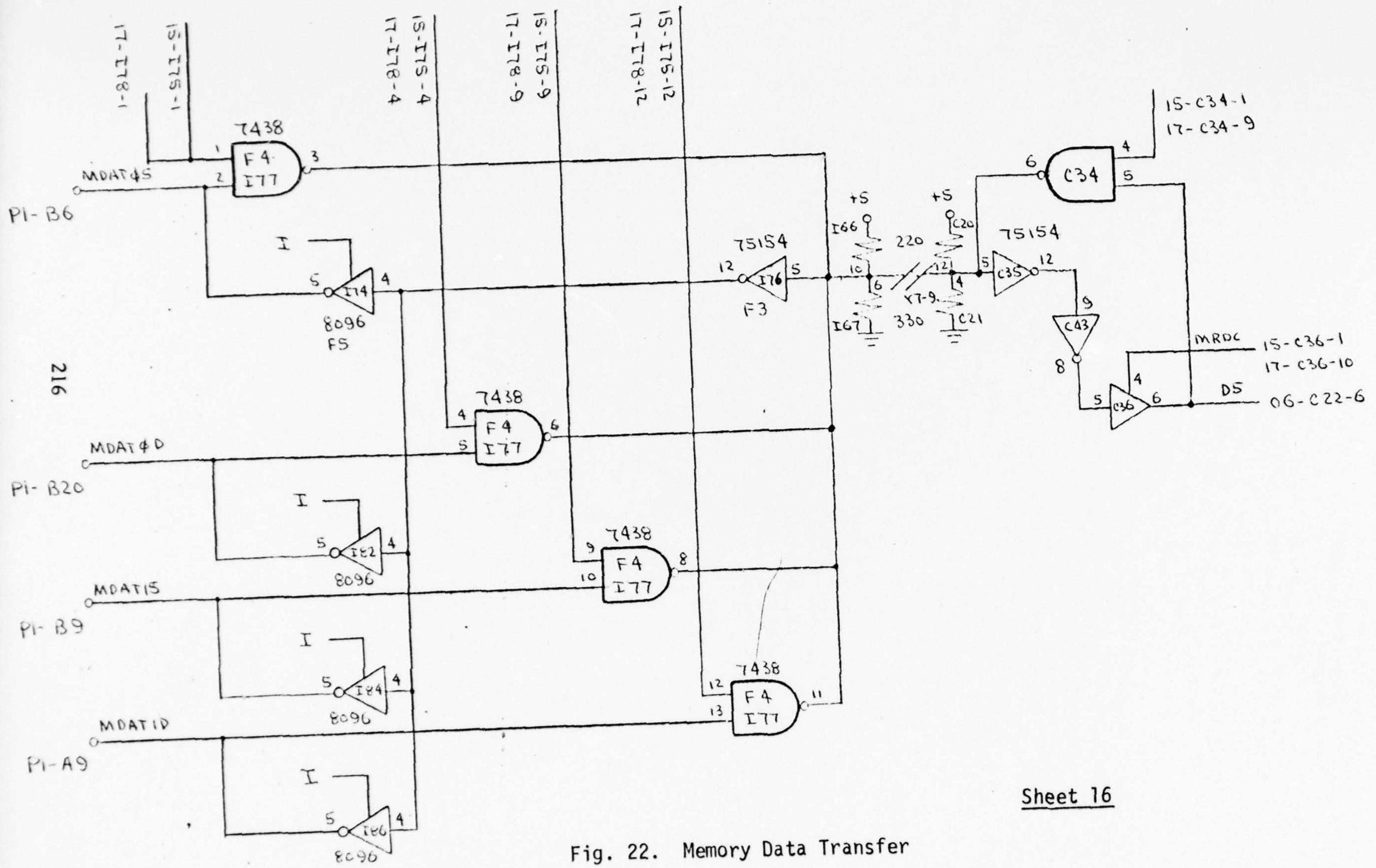
Sheet 14

Fig. F-20. Memory Data Transfer



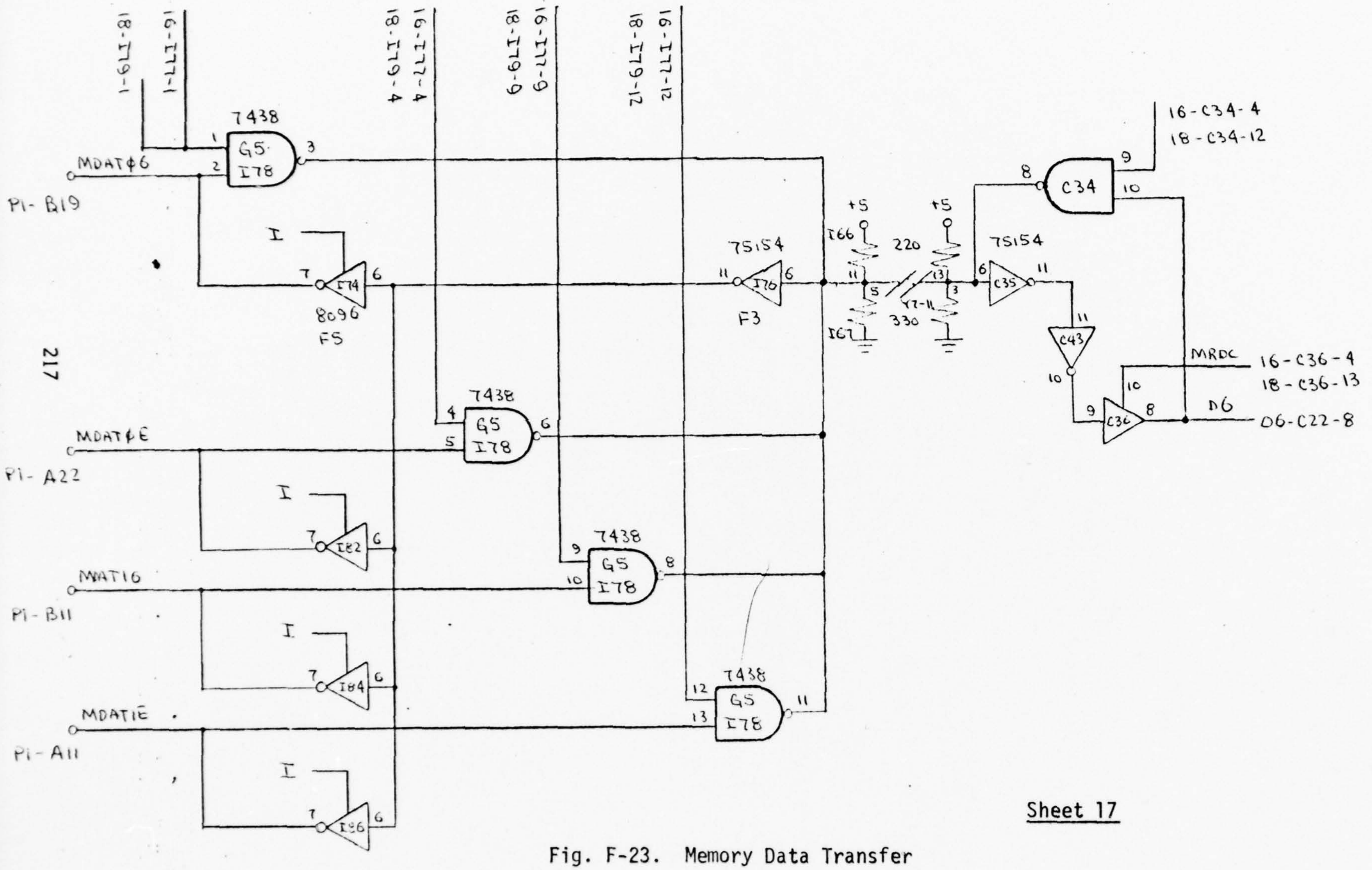
Sheet 15

Fig. F-21. Memory Data Transfer



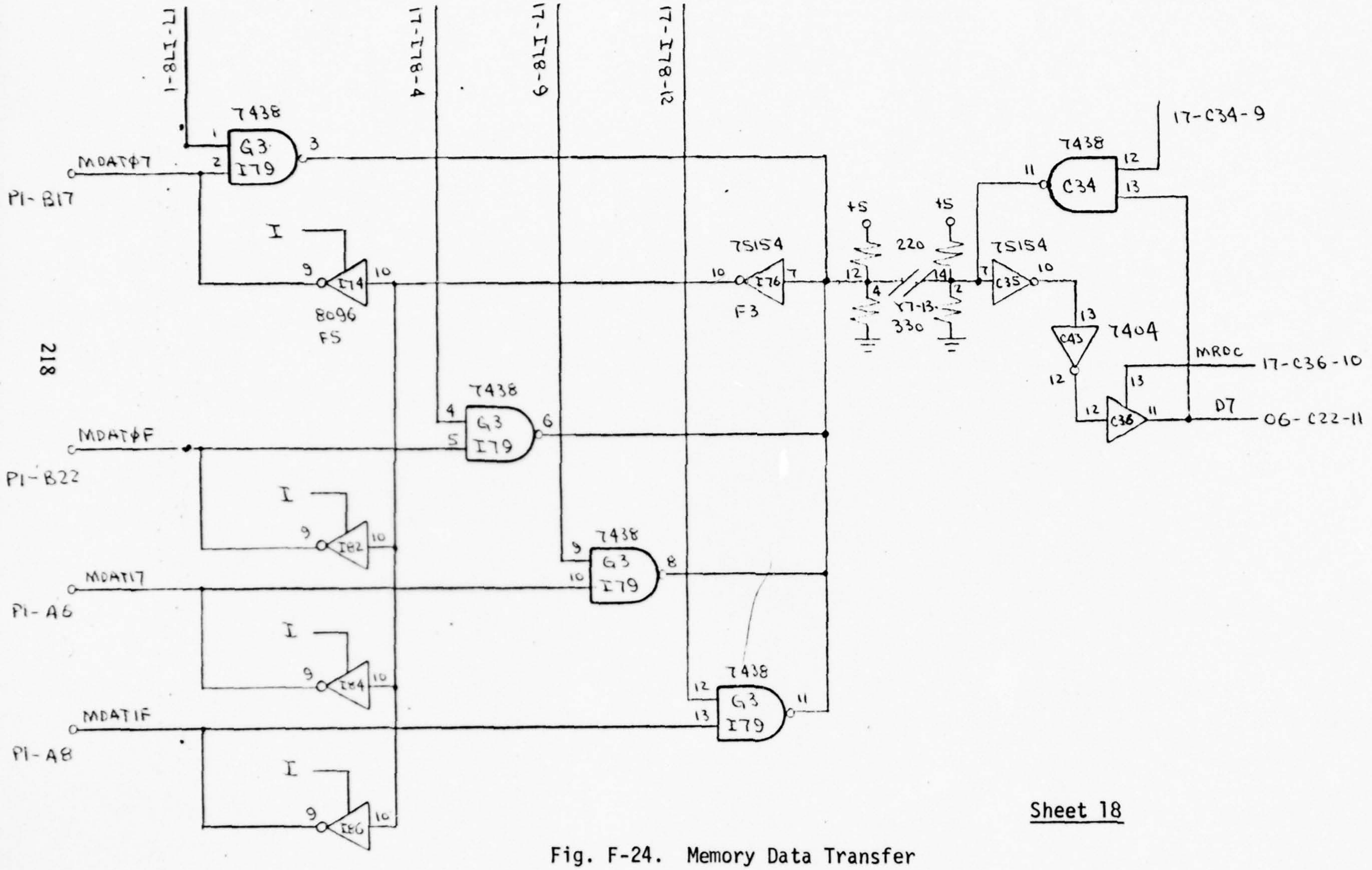
Sheet 16

Fig. 22. Memory Data Transfer



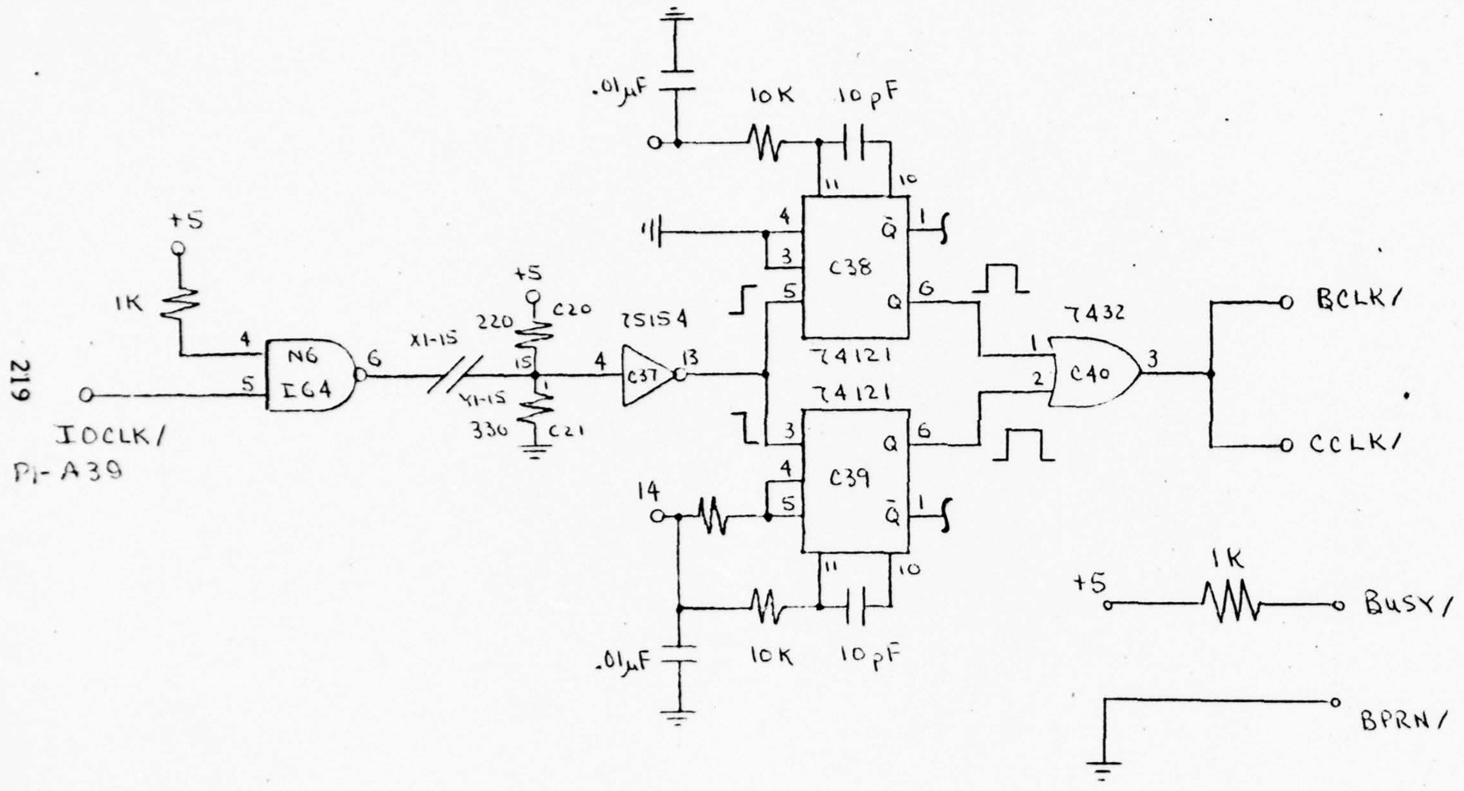
Sheet 17

Fig. F-23. Memory Data Transfer



Sheet 18

Fig. F-24. Memory Data Transfer



Sheet 19

Fig. F-25. Clock Generator

Appendix G

Tri-State Logic

Introduction

Tri-state logic may be referred to, at times, as bus--organized TTL, wire--OR'able TTL, or even bus OR'able logic. The "WIRE-OR" designation is commonly used with this type of concept, although in the case of tri-state logic no Boolean function is achieved. "Tri-State" basically defines a logic element which has three distinct output states: Zero, One (normal TTL levels) and OFF wherein an OFF state represents a high impedance condition which can neither sink nor source current at a definable logic level. At most, it may require 40 uA leakage current to be supplied to it from other devices connected to the same output line, or it may supply up to 40 uA current to another device on the line.

Effectively, then, these devices have all the desirable features of TTL (greater noise immunity, good rise times, line driving capability, etc.) plus the ability to interconnect outputs of similar devices.

Tri-state is useful in many facets of system design. Specifically, for this project design the tri-state logic will allow direct connection to the I/O and memory buses and at the same time insure system compatibility. It becomes obvious immediately that multiplexing (regardless of the function) can be performed right on the bus line. Examples of other feats that can be performed with tri-state logic are high-speed time sharing of decoder-drivers, fast random-access (or sequential) memory arrays, and bi-directional line driving.

Design Implementation

The concept of creating a tri-state TTL device is relatively simple. Essentially, the tri-state device removes the drive current from the totem-pole output of the TTL device. The output then resembles two semiconductor junctions biased in the non-conducting or high impedance state. As stated previously, these junctions do exhibit some leakage which must be provided for in the outputs of the device that is driving the bus line. For this reason all tri-state logic elements are designed to provide 5.2 mA in the logic 1 state (2.4 V minimum). If 128 tri-state outputs are connected to a bus line, the single ON device must supply (worst-case) the maximum leakage of 127 OFF devices and still have current left to drive the receiving TTL elements:

127 x 40 uA = 5.08 mA

5.2 mA - 5.08 mA = 120 mA

The above calculation indicates that three unit TTL loads can be driven even with 127 other devices connected to the line.

In order to provide the high output current required by tri-state connections the TTL output contains a darlington-connected upper stage.

The above essentially describes the basic device design considerations necessary to convert ordinary TTL to tri-state operation. There are some other minor changes but these are necessary primarily for fabriation reasons.

Advantages

Tri-state logic is unusual and therefore generates unusual considerations. For example, because of its higher source current capability, it has two distinct advantages over ordinary TTL. One advantage is that it can drive a greater length of line in line-driving applications. Ordinary TTL, having only 400 uA source capability, can drive approximately 10-12 inches of line before noise becomes a problem. All tri-state devices, however, can source 5.2 mA minimum and therefore can drive over 10 feet of cable reliably.

The other advantage attributable to the high source current capability of tri-state devices is the greatly reduced on-level output impedance which gives much better one-level noise immunity; being approximately a factor of 10 better. It is important to note that these advantages are only side benefits of tri-state. They come with every tri-state device regardless of function.

All tri-state devices behave like totem-pole TTL devices when they are in the ON state. It is necessary therefore that (1) all other devices connected to the same bus line must be OFF (Hi-Z state) and (2) that a convenient method of selecting the ON device in a system be provided.

Finally, it is important to note that 40 uA is all that is required whether the input be a 0 or a 1. The units are designed this way so that an array of devices can be driven from buffer type elements without having to handle the 1.6 mA zero level input current of all the OFF devices all the time. For example, a single DM8093 or DM8094 could drive an input array of greater than 128 devices and yet only have to sink less than 6.6 mA in the 0 state (approximately 5.08 mA for all the OFF devices and 1.6 mA for the ON device). So, in addition to making it easy to fan-out into large number of devices, they are also easy to fan into.

APPENDIX H

Parts List

Appendix H contains only the list of integrated circuits used in the design. A component designator starting with the letter "I" means that it can be found on the wire-wrap board located within the MTC cabinet while a component designator starting with the letter "C" means that it can be found on the wire-wrap board located in the enclosure near the disk drive unit.

A part number starting with "SN" is a Texas Instrument number whose full description and characteristics can be found in TI's _TTL Integrated Circuits Catalog_; "FCD" is a Fairchild number whose full description and characteristics can be found in Fairchild's _TTL Data Book_; "DM" is a National Semiconductor number whose full description and characteristics can be found in National Semiconductor's _The TTL Data Book_.

Component Designator	Mfg Part Number	Description
I1	DM 8093	Tri-State Quad 2-Input Buffer
I2	SN 7440	Dual 4-Input NAND Buffer (O.C.)
I3	SN 7440	Dual 4-Input NAND Buffer (O.C.)
I4	SN 7440	Dual 4-Input NAND Buffer (O.C.)
I5	SN 7404	Dual 4-Input NAND Buffer (O.C.)
I6	SN 7400	Quad 2-Input NAND Gate
I7	FCD 9301	1-of-10 Decoder
I8	FCD 9301	1-of-10 Decoder
I9	SN 7440	Dual 4-Input NAND Buffer (O.C.)
I10	SN 7432	Quad 2-Input OR Gate
I11	DM 8093	Tri-State Quad 2-Input Buffer
I12	DM 8093	Tri-State Quad 2-Input Buffer
I13	DM 8093	Tri-State Quad 2-Input Buffer
I14	SN 7402	Quad 2-Input NOR Gate
I15	SN 7438	Quad 2-Input NAND Gate (O.C.)
I16	SN 74121	Monostable Multivibrator
I17	SN 74121	Monostable Multivibrator
I18	SN 7474	Dual D Flip-Flop
I19	SN 7404	Hex Inverter
I20	--	
I21	SN 7410	Triple 3-Input NAND Gate
I22	SN 74121	Monostable Multivibrator
I23	SN 7474	Dual D Flip-Flop
I24	SN 75154	Quad Line Receiver
I25	916C-221-X2PE	15 Resistors 16-Pin DIP 220 ohm

Component Designator	Mfg Part Number	Description
I26	916C-331-X2PE	15 Resistors 16-Pin DIP 330 ohm
I27	SN 74121	Monostable Multivibrator
I28	FCD 9301	1-of-10 Decoder
I29	DM 8097	Tri-State Hex 2-Input Buffer
I30	DM 8096	Tri-State Hex 2-Input Buffer
I31	SN 7438	Quad 2-Input NAND Gate (O.C.)
I32	SN 75154	Quad Line Receiver
I33	--	
I34	SN 7438	Quad 2-Input NAND Gate (O.C.)
I35	SN 7438	Quad 2-Input NAND Gate (O.C.)
I36	DM 8097	Tri-State Hex 2-Input Buffer
I37	DM 8096	Tri-State Hex 2-Input Buffer
I38	SN 7438	Quad 2-Input NAND Gate (O.C.)
I39	SN 75154	Quad Line Receiver
I40	SN 7404	Hex Inverter
I41	FCD 9314	4-BitLatch
I42	SN 7486	Quad Exclusive OR Gate
I43	DM 8093	Tri-State Quad 2-Input Buffer
I44	SN 7438	Quad 2-Input NAND Gate (O.C.)
I45	SN 7425	Dual 4-Input NOR Gate
I46	SN 7438	Quad 2-Input NAND Gate (O.C.)
I47	SN 7438	Quad 2-Input NAND Gate (O.C.)
I48	SN 7438	Quad 2-Input NAND Gate (O.C.)
I49	SN 75154	Quad Line Receiver
I50	SN 75154	Quad Line Receiver

Component Designator	Mfg Part Number	Description
I51	SN75154	Quad Line Receiver
I52	SN 7404	Hex Inverter
I53	SN 7404	Hex Inverter
I54	916C-221-X2PE	15 Resistors 16-Pin DIP 220 ohm
I55	916C-331-X2PE	15 Resistors 16-Pin DIP 330 ohm
I56	SN 74121	Monostable Multivibrator
I57	SN 7407	Hex Driver
I58	--	
I59	SN 7474	Dual D Flip-Flop
I60	SN 75154	Quad Line Receiver
I61	SN 7404	Hex Inverter
I62	FCD 9321	Dual 1-of-4 Decoder
I63	SN 7402	Quad 2-Input NOR Gate
I64	SN 7438	Quad 2-Input NAND Gate (O.C.)
I65	SN 7438	Quad 2-Input NAND Gate (O.C.)
I66	916C-221-X2PE	15 Resistors 16-Pin DIP 220 ohm
I67	916C-331-X2PE	15 Resistors 16-Pin DIP 330 ohm
I68	DM 8096	Tri-State Hex 2-Input Buffer
I69	SN 7438	Quad 2-Input NAND Gate (O.C.)
I70	SN 75154	Quad Line Receiver
I71	SN 7438	Quad 2-Input NAND Gate (O.C.)
I72	SN 7438	Quad 2-Input NAND Gate (O.C.)
I73	SN 7438	Quad 2-Input NAND Gate (O.C.)

Component Designator	Mfg Part Number	Description
I74	DM 8096	Tri-State Hex 2-Input Buffer
I75	SN 7438	Quad 2-Input NAND Gate (O.C.)
I76	SN 75154	Quad Line Receiver
I77	SN 7438	Quad 2-Input NAND Gate (O.C.)
I78	SN 7438	Quad 2-Input NAND Gate (O.C.)
I79	SN 7438	Quad 2-Input NAND Gate (O.C.)
I80	SN 7438	Quad 2-Input NAND Gate (O.C.)
C1	DM 8094	Tri-State Quad 2-Input Buffer
C2	DM 8094	Tri-State Quad 2-Input Buffer
C3	SN 75154	Quad Line Receiver
C4	SN 75154	Quad Line Receiver
C5	916C-221-X2PE	15 Resistors 16-Pin DIP 220 ohm
C6	916C-331-X2PE	15 Resistors 16-Pin DIP 330 ohm
C7	DM 8093	Tri-State Quad 2-Input Buffer
C8	SN 7438	Quad 2-Input NAND Gate (O.C.)
C9	SN 7404	Hex Inverter
C10	SN 7438	Quad 2-Input NAND Gate (O.C.)
C11	SN 75154	Quad Line Receiver
C12	DM 8094	Tri-State Quad 2-Input Buffer
C13	SN 75154	Quad Line Receiver
C14	FCD 8551	Tri-State 4-Bit Latch
C15	FCD 8551	Tri-State 4-Bit Latch
C16	SN 74121	Monostable Multivibrator
C17	SN 7474	Dual D Flip-Flop

Component Designator	Mfg Part Number	Description
C18	SN 7438	Quad 2-Input NAND Gate (O.C.)
C19	SN 75154	Quad Line Receiver
C20	916C-221-X2PE	15 Resistors 16-Pin DIP 220 ohm
C21	916C-331-X2PE	15 Resistors 16-Pin DIP 330 ohm
C22	DM 8094	Tri-State Quad 2-Input Buffer
C23	DM 8094	Tri-State Quad 2-Input Buffer
C24	DM 8094	Tri-State Quad 2-Input Buffer
C25	DM 8094	Tri-State Quad 2-Input Buffer
C26	SN 7438	Quad 2-Input NAND Gate (O.C.)
C27	SN 7438	Quad 2-Input NAND Gate (O.C.)
C28	SN 7438	Quad 2-Input NAND Gate (O.C.)
C29	SN 7438	Quad 2-Input NAND Gate (O.C.)
C30	SN 7438	Quad 2-Input NAND Gate (O.C.)
C31	SN 7438	Quad 2-Input NAND Gate (O.C.)
C32	SN 75154	Quad Line Receiver
C33	DM 8094	Tri-State Quad 2-Input Buffer
C34	SN 7438	Quad 2-Input NAND Gate (O.C.)
C35	SN 75154	Quad Line Receiver
C36	DM 8094	Tri-State Quad 2-Input Buffer
C37	SN 75154	Quad Line Receiver
C38	SN 74121	Monostable Multivibrator
C39	SN 74121	Monostable Multivibrator
C40	SN 7432	Quad 2-Input OR Gate
C41	SN 7404	Hex Inverter
C42	SN 7404	Hex Inverter
C43	SN 7404	Hex Inverter

Component Designator	Mfg Part Number	Description
I81	DM8096	Tri-State Hex Buffer
I82	DM8096	Tri-State Hex Buffer
I83	DM8096	Tri-State Hex Buffer
I84	DM8096	Tri-State Hex Buffer
I85	DM8096	Tri-State Hex Buffer
I86	DM8096	Tri-State Hex Buffer

APPENDIX I

Recording Format

This appendix summarizes the specifications for the IBM soft-sectored recording format used by the Diskette Operating System.

Physical Data Format

The physical data format is the format that the diskette controller circuitry must interact with. The elements of the physical data format are the hard index hole, index mark, sector address marks, sector headers, and data sectors. The index mark and sector address marks are recorded with unique clock patterns requiring the controller circuitry to accumulate the unique clock patterns for index and sector address mark identification. Figure I–1 illustrates the general physical data format.

A "byte," when referring to serial data (being written to or read from the diskette drive), is defined as eight (8) consecutive bit cells. The most significant bit cell is defined as bit cell 0 and the least significant bit cell is defined as bit cell 7. When reference is made to a specific data bit (i.e., data bit 3), it is with respect to the corresponding bit cell (bit cell 3).

During a write operation bit cell 0 of each byte is transferred to the drive first with bit cell 7 being transferred last. Correspondingly, the most significant byte of data is transferred to the diskette first and the least significant byte is transferred last.

When data is being read back from the drive, bit cell 0 of each byte will be transferred first with bit cell 7 last. As with reading, the most significant byte will be transferred first from the drive to the user.

Figure I-2 illustrates the relationship of the bits within a byte and Figure I–3 illustrates the relationship of the bytes for read and write data.

Data is recorded on the diskette using frequency modulation as the recording mode; that is, each data bit recorded on the diskette has an associated clock bit recorded with it. Data written on and read back from the disc takes the form shown in Figure I-4.

As shown in Figure I-5, the clock bits and data bits (if present) are interleaved. By definition, a Bit Cell is the period between the leading edge of one clock bit and the leading edge of the next clock bit.

Track Format

Each track recorded on a diskette consists of 26 fixed length records along with the necessary gaps for record updating. Figure I-6 illustrates the format of one complete track.

Each field on a track is separated from adjacent fields by a number of bytes containing no data. These areas are referred to as gaps and are provided to allow the updating of one field without affecting adjacent fields. As can be seen from Figure I-6, there are four different types of gaps on each track:

- Gap 1 - Post-Index Gap

  This gap is defined as the 32 bytes between Index Address Mark and the ID Address Mark for Sector one (excluding the address mark bytes). This gap is always 32 bytes in length and is not affected by any updating process.

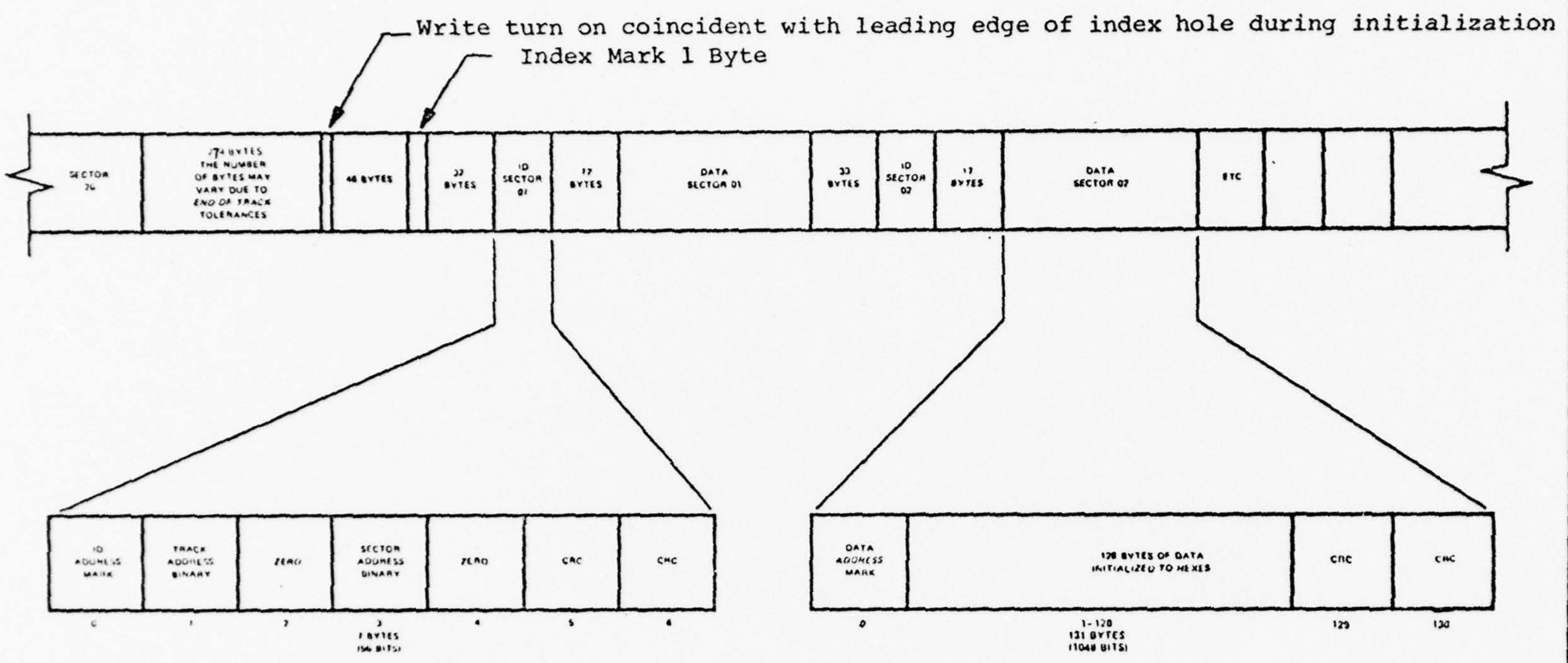


Fig. I-1. Physical Data Format

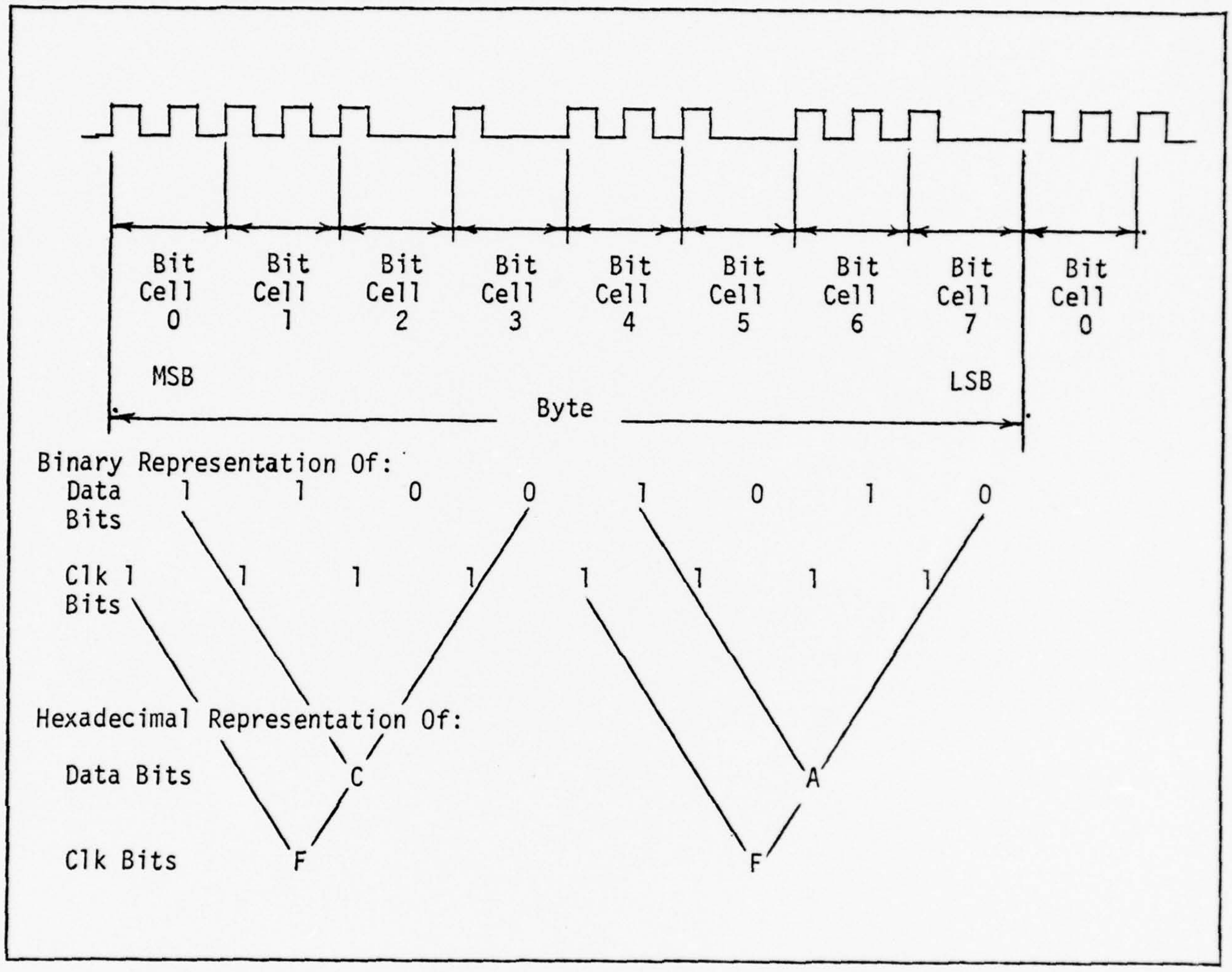
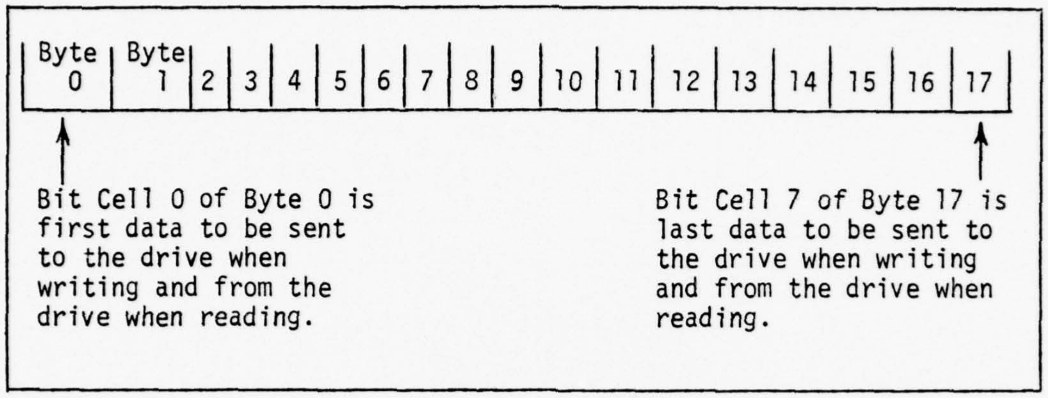


Fig. I-2. Byte Representation



Byte 0
Byte 1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
Bit Cell 0 of Byte 0 is first data to be sent to the drive when writing and from the drive when reading.
Bit Cell 7 of Byte 17 is last data to be sent to the drive when writing and from the drive when reading.

Fig. I-3. Data Bytes

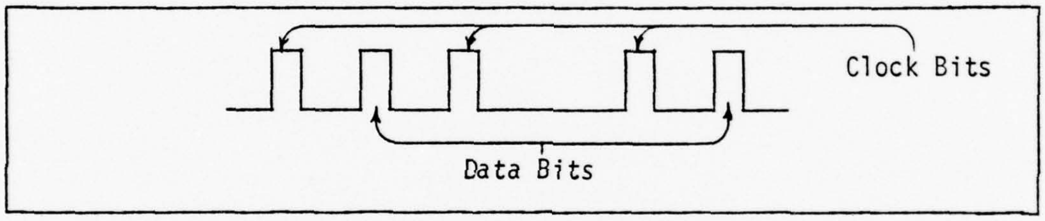


Fig. I-4. Data Bit

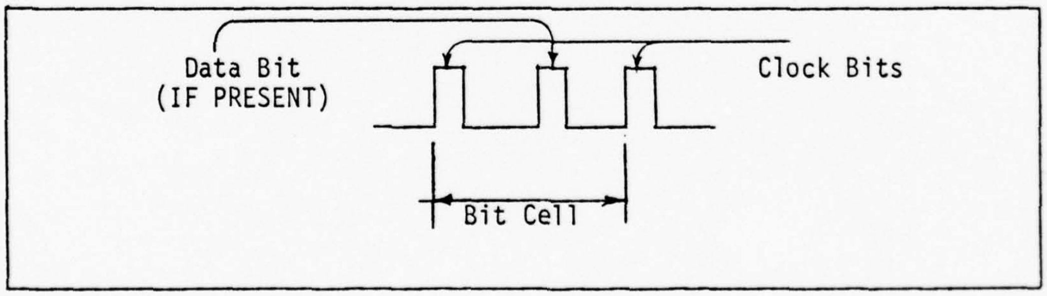


Fig. I-5. Bit Cell

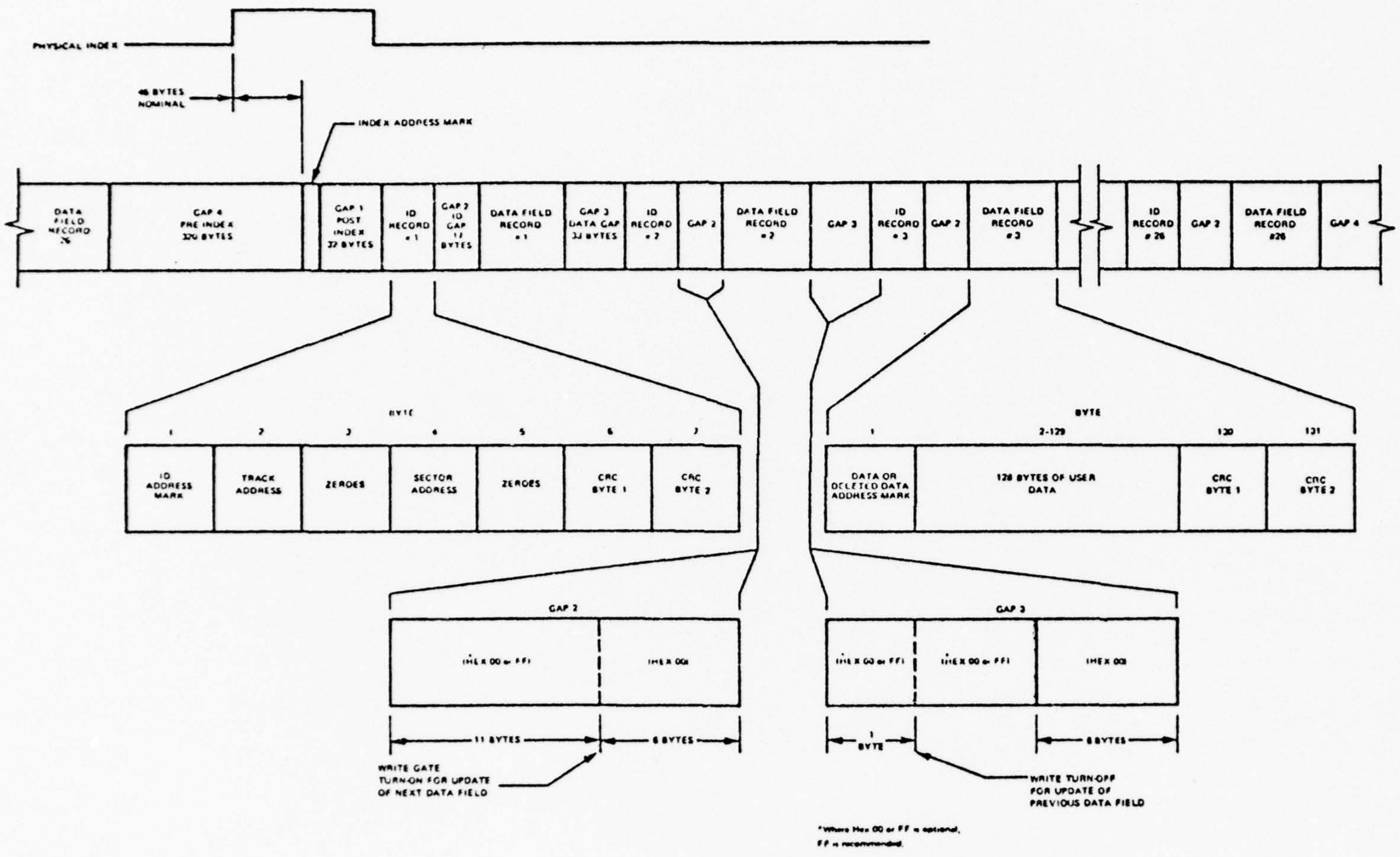


Fig. I-6. Track Format

- Gap 2 - ID Gap

  The seventeen bytes between the ID Field and the Data Field are defined as Gap 2 (ID Gap). This gap may vary in size slightly after the Data field has been updated.

- Gap 3 - Data Gap

  The thirty-three bytes between the Data field and the next ID field are defined as Gap 3 (Data Gap). As with the ID Gap, the Data Gap may vary slightly in length after the adjacent Data field has been updated.

- Gap 4 - Pre-Index Gap

  The three-hundred and twenty bytes between the last Data field on a track and the Index Address Mark are defined as Gap 4 (Pre-Index Gap). Initially, this gap is nominally 320 bytes in length; however, due to write frequency tolerances and diskette speed tolerances this gap may vary slightly in length. Also, after the data field of record 26 has been updated this gap may again chance slightly in length.

Address Marks

Address Marks are unique bit patterns one byte in length which are used to identify the beginning of ID and Data fields and to synchronize the deserializing circuitry with the first byte of each field. Address Mark bytes are unique from all other data bytes in that certain bit cells do not contain a clock bit (all other data bytes have clock bits in every bit cell). There are four different types of Address Marks used. Each of these is used to identify different types of fields.

- Index Address Mark

  The Index Address Mark is located at the beginning of each track and is a fixed number of bytes in front of the first record. The bit configuration for the Index Address Mark is shown in Figure I-7.

- ID Address Mark

  The ID Address Mark byte is located at the beginning of each ID field on the diskette. The bit configuration for this Address Mark is shown in Figure I-8.

- Data Address Mark

  The Data Address Mark byte is located at the beginning of each non-deleted Data Field on the diskette. The bit configuration for this Address Mark is shown in Figure I-9.

- Deleted Data Address Mark

  The Deleted Data Address Mark byte is located at the beginning of each deleted Data field on the diskette. The bit configuration for this Address Mark is shown in Figure I-10.

CRC Bytes

Each field written on the diskette is appended with two Cyclic Redundancy Check (CRC) bytes. These two CRC bytes are generated from a cyclic permutation of the data bits starting with bit zero of the address mark and ending with bit seven of the last byte within a field (excluding the CRC bytes). This cyclic permutation is the remainder from the division of the data bits in the field (represented as an algebraic polynomial) by a generator polynomial G. For all fields recorded on a diskette, this generator polynomial is:

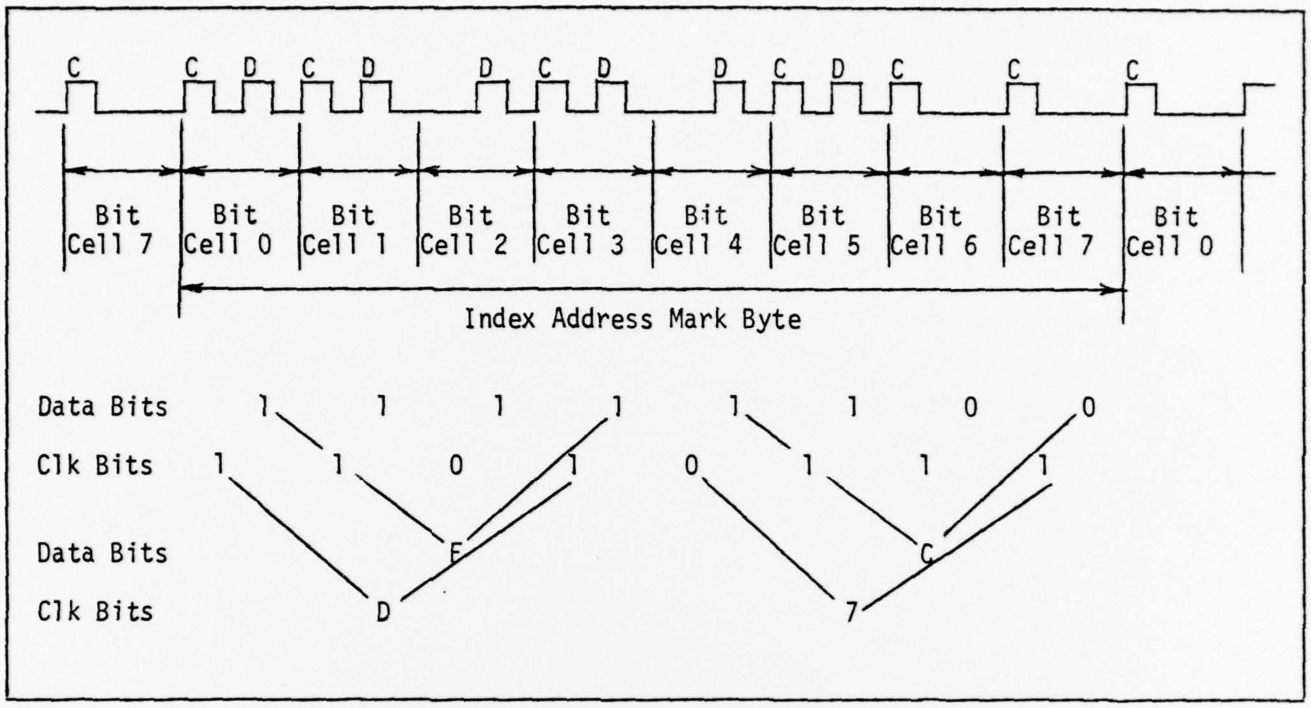


Fig. I-7. Index Address Mark

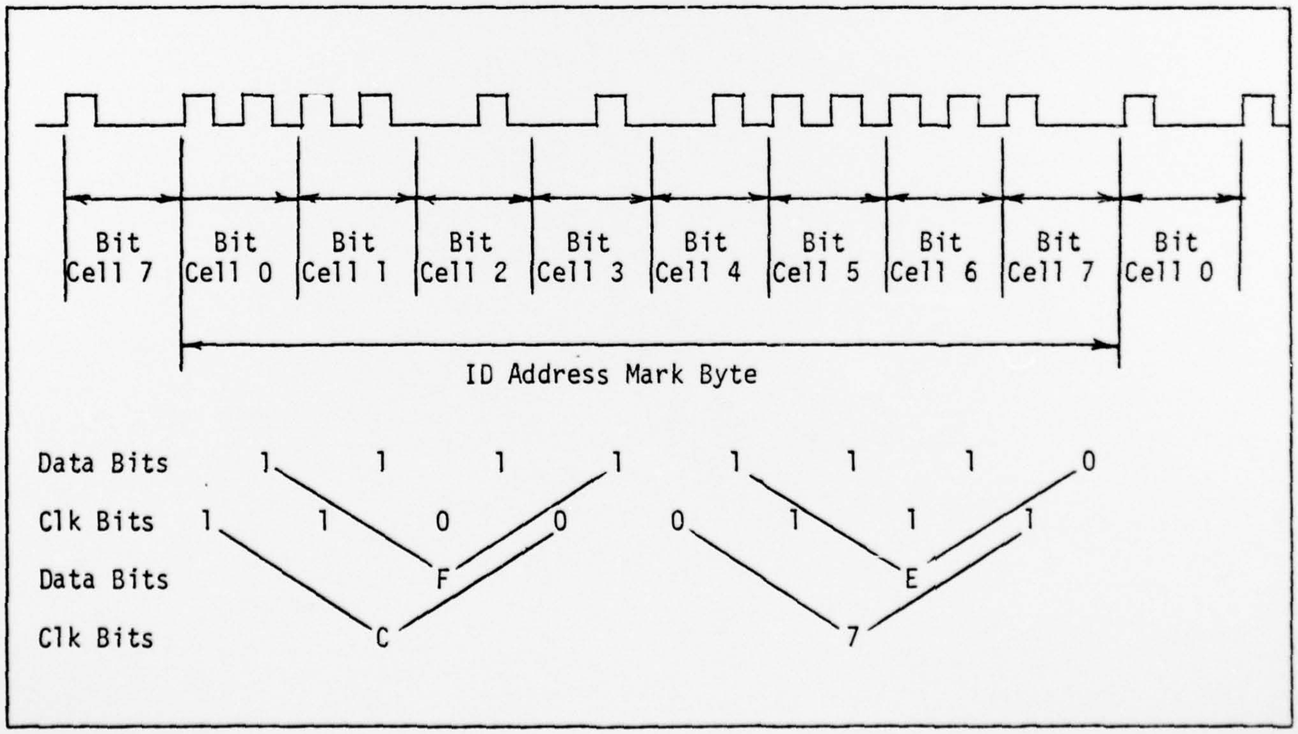


Fig. I-8. ID Address Mark

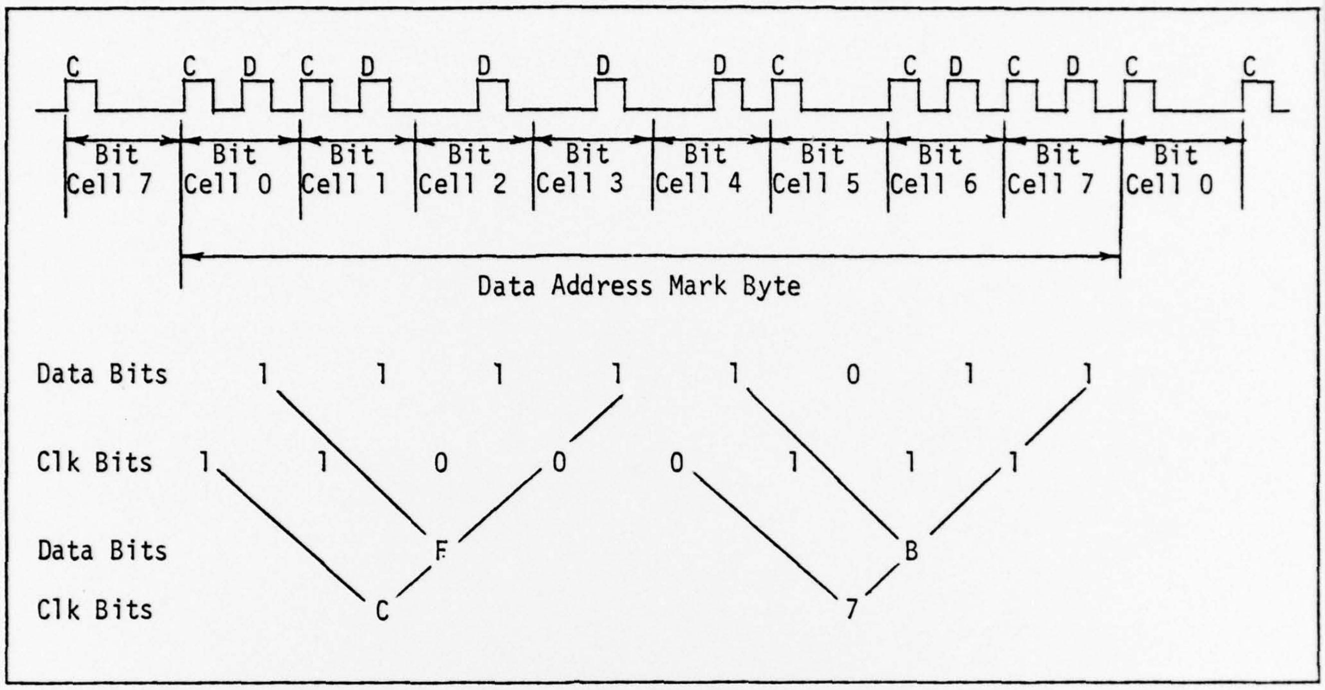


Fig. I-9. Data Address Mark

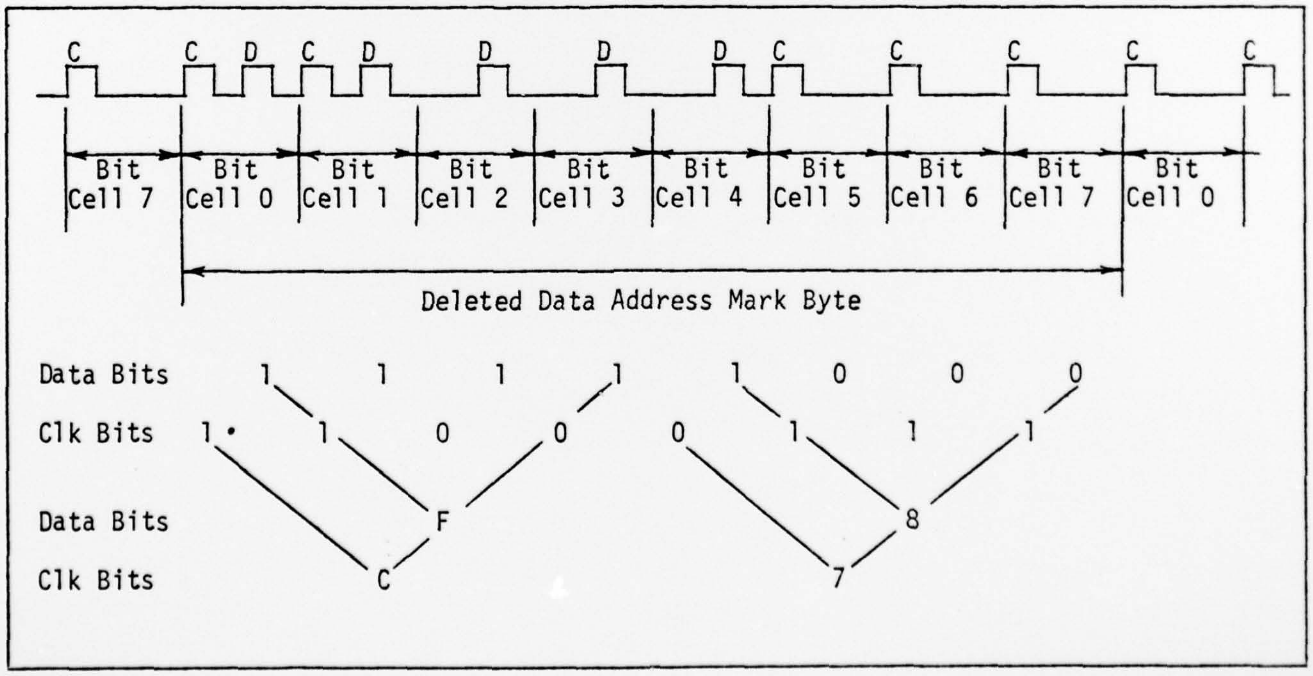


Fig. I-10. Deleted Data Address Mark

$$G(X) = X^{16} + X^{12} + X^{5} + 1$$

When a field is read back from a diskette, the data bits (from bit zero of the address mark to bit seven of the second CRC byte) are divided by the same generator polynomial G(X) and a non-zero remainder indicates an error within the data read back from the drive while a remainder of zero indicates the data has been read back correctly from the diskette or an undetectable error has been read back.

Appendix J

Wiring Lists

The wiring lists for the MTC Interface Board, Test Board, and Tx/Rx Logic Board have been given to the project office (AFLC/ACTAA) by special memorandum.

Appendix J does contain, however, the critical test points located on the MTC test point extension (see Figure 6-4) and the wired pins of the P2, P3, P4, and P5 edge connectors (see Figure 6-8).

Channel Board Connector P3	Interface Board Connector P4	Tx/Rx Logic Board Connector P2	Test Board Connector P5	Signal Mnemonic (FDD)
A1	1			Grd
A2	3			+5
A3	5			+5
A4	7			+12
A5	9			Spare
A6	11			Grd
A7	13	2		BCLK/
A8	15	50		Grd
A9	17	51		HPR
A10	19	92	L	MRDC/
A11	21	86		IORC/
A12	23	88	N	XACK/
A13	25			Special
A14	27			Not Used

Channel Board Connector P3	Interface Board Connector P4	Tx/Rx Logic Board Connector P2	Test Board Connector P5	Signal Mnemonic (FDD)
A15	29			Not Used
A16	31	62		CCLK/
A17	33			Special
A18	35			INT6
A19	37			INT4
A20	39			INT2
A21	41			INT1/
A22	43	28	9	ADRE/
A23	45	22	R	ADRC/
A24	47	75	T	ADRA/
A25	49	68	U	ADR8/
A26	51	27	V	ADR6/
A27	53	82	W	ADR4/
A28	55	14	X	ADR2/
A29	57	8	Y	ADR0/
A30	59			DATE/
A31	61			DATC/
A32	63			DATA/
A33	65			DAT8/
A34	67	108	J	DAT6/
A35	69	44	F	DAT4/
A36	71	100	D	DAT2/
A37	73	38	B	DAT0/
A38	75			Grd

Channel Board Connector P3	Interface Board Connector P4	Tx/Rx Logic Board Connector P2	Test Board Connector P5	Signal Mnemonic (FDD)
A39	77			-10
A40	79			-12
A41	81			+5
A42	83			+5
A43	85			Grd
B1	4			Grd
B2	4			+5
B3	6			+5
B4	8			+12
B5	10			Spare
B6	12			Grd
B7	14	83		INIT/
B8	16			BPRO/
B9	18			BREQ/
B10	20	93	M	MWTC/
B11	22	87		IOWC/
B12	24			INH1
B13	26			INH2
B14	28			Spare
B15	30			Spare
B16	32			Spare
B17	34			Spare
B18	36			INT7
B19	38			INT5/
B20	40	74	P	INT3/

Channel Board Connector P3	Interface Board Connector P4	Tx/Rx Logic Board Connector P2	Test Board Connector P5	Signal Mnemonic (FDD)
B21	42			INT1/
B22	44	31	7	ADRF/
B23	46	25	S	ADRD/
B24	48	80	11	ADRB/
B25	50	10	13	ADR9/
B26	52	32	15	ADR7/
B27	54	26	17	ADR5/
B28	56	20	19	ADR3/
B29	58	70	21	ADR1/
B30	60			DATF/
B31	62			DATD/
B32	64			DATB/
B33	66			DAT9/
B34	68	48	K	DAT7/
B35	70	104	H	DAT5/
B36	72	40	E	DAT3/
B37	74	98	C	DAT1/
B38	78			Grd
B39	78			-10
B40	80			-12
B41	82			+5
B42	84			+5
B43	86			Grd

Test Points

Mnemonic	MTC Board Location	TP-Extension Location
Func2	L6-3	S-2
Fun2/	M3-8	S-3
Func1	L6-6	S-4
Func1/	N3-8	S-5
Func0	L6-8	S-6
Func0/	P3-8	S-7
Output EN/	K5-3	T-1
Inout EN/	K5-6	T-2
F=1	L5-12	T-3
F=2	L5=11	T-4
F=3	L5-10	T-5
F=6 ECK0/	L5-4	T-6
F=7	L5-5	T-7
F=0	L4-13	U-1
F=1	L4-12	U-2
F=3	L4-10	U-3
F=6 ECKI/	L4-4	U-4
F=7 CCREQ/	L4-5	U-5
IOFSEL	N4-3	U-6
DEVEN	R2-6	S-1
IORESET	G4-6	U-13
DEL3I	J2-3	U-7
DEL30	MI-6	V-1

Mnemonic	MTC Board Location	TP-Extension Location
DEL1I	J3-3	V-2
DEL10	K3-6	V-3
DEL2I	J4-3	V-4
DEL20	K3-8	V-5
DATIEN	N5-4	V-6
DATOEN	C5-8	V-7
ADROEN	M5-8	W-1
IODAT71	A2-3	W-2
IODAT6/	A2-6	W-3
IODAT5/	A2-8	W-4
IODAT4/	A2-11	W-5
IODAT3/	B2-3	W-6
IODAT2/	B2-6	W-8
IODAT1/	B2-8	W-9
IODAT0/	B2-11	W-10
IOCLK	N6-6	W-7
GAITAD/	M6-8	W-11
CIREQ/	M6-12	W-12
INTDAT3	J1-10	W-13
INTEN1/	N4-6	W-14
INT	M1-11	V-8
CLR CIREQ/	M1-13	V-9
MWTEN/	R1-13	V-10
MRDEN/	R1-12	V-11
WTBYTEN	N5-10	V12

Mnemonic	MTC Board Location	TP-Extension Location
WTEN/	N5-9	V-13
ADRIEN1	C5-4	V-14
XACKEN/	N4-8	U-8
WRTBYT0 EN/	L3-4	U-9
RBYTEN0	K4-13	U-10
ADR E	L3-2	U-11
ADR F	L3-3	U-12

VITA

Michael James Varner was born on 6 August 1950 in Sidney, Nebraska. He graduated from Williston Senior High School, Williston, North Dakota, in 1968. He then attended North Dakota State University, Fargo, North Dakota from which he received his Bachelor of Science in Electrical and Electronics Engineering and a commission in the United States Air Force in May 1972. Subsequent to his commissioning, he attended the Communications-Electronics Engineering School at Keesler AFB, Mississippi for a period of six months. Before coming to the Air Force Institute of Technology in September 1975, he served two and one-half years at Scott AFB, Illinois as the Maintenance Control Officer for the 1918th Communications Squadron, now the 1974th Communications Group. He is married to the former Judy Irene Lundy and presently has two sons, Michael Wain and Eric James.

Permanent Address: P.O. Box 338
Billings, Montana 59101

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER GCS/EE/77-2 ✓	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DESIGN OF A FLOPPY DISK DRIVE CONTROLLER INTERFACE FOR THE 65070-11 MULTIPLE TERMINAL CONTROLLER		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) MICHAEL J. VARNER, Captain, USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology ✓(AFIT-EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Performance Analysis Branch AFLC/ACTAA Wright-Patterson AFB, Ohio 45433		12. REPORT DATE 18 March 1977 13. NUMBER OF PAGES 248
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

APPROVED FOR PUBLIC RELEASE, IAW AFR 190-17.

[illegible]
JERRAL F. GUESS, Capt, USAF
Director of Information

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Floppy Disk Drive
Multiple Terminal Controller
Interface

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The 1970's was to be the decade when AFLC would completely automate its transactions on a "real time" basis through a program called the Advanced Logistics System. However, the system was never completed due to implementation difficulties. A part of that system was the Multiple Terminal Controller, which is a minicomputer, but which lacks storage capability to be used for most Air Force applications. Based on the desire

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

to use the Multiple Terminal Controller as a stand-alone system, this report develops an interface design for a floppy disk drive which will give the MTC secondary-storage and hence a stand-alone capability.

The design of the interface is presented which encompasses the three general functions of I/O control, direct-memory access, and program interrupt. The design is fully documented in register transfer language and other logic expressions which are finally used to realize and implement the design. While the software used to test the design is presented, the report does not discuss the software required to integrate the design into an operating system.